Improving Machine Learning Models for Large Data and Image Detection

By

ROBIN ELIZABETH YANCEY
DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

____

Norm Matloff, Chair

____

Richard Levenson

____

Joshua McCoy

Committee in Charge

2022

Improving Machine Learning Models for Large Data and Image Detection

The combination of Big Data with the memory size limitations common in the moderately-sized machines used in many complex ML applications motivates a search for fast parallel computation methods. In this work, a method called *Software Alchemy*, a technique to develop *generally applicable* parallel ML methods, while avoiding the need for a different approach to every algorithm will be demonstrated.

Note, that this a multi-topic dissertation. Imaging processing is another major area where there is an increasing amount of data to deal with. Classic methods of image fraud detection often require prior knowledge of the method of forgery in order to determine which features to extract from the image to localize the region of interest. Thus, ***as the second main topic***, certain classic and newer machine-learning based techniques of image fraud detection will be modified, combined, or parallelized to produce improved results compared to the classic, stand-alone, serial technique.

Machine learning methods outperform the traditional methods by quickly and automatically extracting a combination of complex/hierarchical features. However, most of these new methods are basic CNN modifications with various filters added (targeting a type of forgery), which still lack the ability to be generalized to a wide range of forgery types and the ability to localize the tampered region. Consequently, object detection frameworks will then be adapted to better solve these problems by extracting various complex features within the localized region. These will out-perform both state-of-the-art classic techniques as well as those in machine learning.

Breast cancer grading is crucial to assessing the stage of breast cancer development, the most common cancer among women, accounting for 50% diagnoses. Yet, it requires a tremendous amount of acquired skill and time for a pathologist to extract the relevant features from many histopathology images by hand to make an assessment. Therefore, ***as the third third main topic***, this object detection technique will be extended to the application of mitosis counting which historically has been done highly similar classic automatic extraction methods. Finally, the newest and most state-of-the-art methods of object detection will be adapted to mitosis counting for the first time to substantially out-perform all other methods.

Contents

0 Overall Introduction

This is a multi-topic dissertation, so the body of work consists of three main topics discussed in Sections 1, 2, and 3. These topics are somewhat unrelated but they do tie together in a number of ways. The major theme of the work is the improvement of machine learning algorithms to make them more efficient than the state-of-the-art methods in terms of both speed and/or accuracy. Each of the frameworks that will be employed will also make use of parallel processing on one or more GPUs, CPUs, or CPU cores for the improvement the speed.

For example, this is the main focus in Section 1, where a technique called Software Alchemy (SA) is used speed to common machine learning algorithms. In this unique form of parallel processing, the results of an algorithm are evaluated on each node before averaging each of the resulting predictions. This is useful in the increasingly common situation in the ML or Big Data age where it is too large to process at all or process quickly on a single machine.

Mainly due to their multi-dimensional nature, images are another form of large data. These are also multiplying in number with the advancements in imaging devices and the number of current social/medical concerns they are used to address. Therefore, for the remainder of the dissertation only image data is used for the optimization and improvement of automatic image tasks. In particular, Section 2 focuses on image fraud detection, while in Section 3 mitosis detection for cancer grading. In Section 2, techniques used to to speed up the slow parts of algorithms are employed by breaking the image into chunks (using SA in some cases), while combinations of different parts of current algorithms are used to make the results more accurate.

In Section 3, the highest performing technique optimized in 2 is adapted to mitosis counting. In either application, this uses a deep learning network on a cloud GPU for detection of the localized region of interest. This outperforms both the slower in-comprehensive methods such as those using traditional manual feature extraction or a newer CNN-only-based method by focusing on the refined feature maps of interest.

Finally, as demonstrated in Section 1, multiple GPU's can also be used to speed up deep learning algorithms. In Section 3, this is employed with a real-time deep learning network for mitosis counting, for the most superior results. Using a combination of data augmentations and an optimized real-time network, this outperforms any other method in literature.

1 Parallelizing Machine Learning Algorithms

1.1 Abstract

The size of the datasets used today and complexity of many ML algorithms, motivates a search for fast parallel computation methods. A further motivating factor is a need to deal with memory size limitations, especially for the moderately-sized machines common in many ML applications. For example, in large genomics studies, recommender

systems, and salary prediction by micro-region and occupation the time complexity for some models can grow as fast as $n^{1.5}$. Even worse, the computation may not fit into available memory in a single machine, rendering the complete estimation nearly impossible. In addition, it is desirable to develop *generally applicable* methods, rather than needing to develop a different parallel approach for every ML algorithm. Here, a technique called *Software Alchemy* will be applied to ML to address these concerns, by breaking the dataset into chunks, applying the given algorithm separately to each chunk.

1.2 Introduction and Significance

One of the main challenges with algorithms of the current era of Big Data is the high cost of added computational complexity with the increase in volume and scale [17]. For example, the support vector machine algorithm (SVM), in terms of the number of data points n, has a $O(n^3)$ time. Moreover, it needs $O(n^2)$ space, forcing a a need to deal with memory size limitations, especially for the moderately-sized machines common in many machine learning (ML) applications. And such limitations arise frequently with Graphics Processing Units.

1.2.1 Software Alchemy

In this work, a technique called *Software Alchemy* (SA) is applied to ML algorithms. The term was named after the fact that SA can ”alchemically” convert algorithms that are generally not *embarrassingly parallel* into embarrassingly parallel ones. This approach is highly preferable because it is widely (if not universally) applicable. One first breaks the dataset into chunks, applying the given algorithm separately to each chunk. The final predicted value is obtained by averaging the results or by “voting” in the classification case [107] [17]. Let’s take an ordinary logistic model, as an example. It would be applied to the data for each chunk, producing estimated coefficients $\widehat{\beta}_{ij}$, where j is the chunk number. The predicted values $\widehat{\beta}_{ij}$ are averaged over all chunks, yielding the final estimated coefficients $\widehat{\beta}_i$. In parametric cases like this, we can achieve the same accuracy with a much lower run time.

1.2.2 Asymptotic Efficiency

It can be shown that chunks averaging is asymptotically of the same statistical accuracy as the full (i.e. unchunked) estimator [107]. In other words, if the sample parameter $\hat{\theta}$ is asymptotically normally distributed, then SA is fully efficient. If the data chunks are i.i.d. vectors $X_1, X_2, ... X_n$, then the chunked estimator $\tilde{\theta}$ will have the same asymptotic variance as the full estimator. (See Section 3 of [107] for the full derivation of this, using characteristic functions.)

1.2.3 Modification for ML

The original form of SA, involving chunks averaging, must be slightly modified for classification types of ML algorithms. In this case, we would either average the estimated conditional class probabilities or choose the most common class overall. When we choose the most common class, this modification of SA will be referred to as *V-SA* (Voting Software Alchemy).

1.2.4 Superlinear Speedup

A major virtue of SA is that *super-linear* speedup can be obtained. This means that a speedup of greater than p for p processes [107] [101] in some applications. This super-linearity is algorithmic in nature, rather than due to, say, cache effects as is typical [36]. For example, if the complexity of the algorithm is $O(n^d)$, and we let r denote the number of parallel processes, then the complexity will be approximately $O[(n/r)^d] = O(n^d/r^d)$, with a speedup of about r^d. If the dimension of the algorithm is greater than one ($d > 1$), we can achieve super-linear speedup, a rarely seen outcome in the parallel computation world. [1] [2]

1.2.5 Memory Issues

One of the main challenges with algorithms in the current era of Big Data is the high cost of added computational complexity with the increase in volume and scale [17]. For example, the support vector machine algorithm (SVM), in terms of the number of data points n, has a $O(n^3)$ time. Moreover, it needs $O(n^2)$ space, forcing a a need to deal with memory size limitations, especially for the moderately-sized machines common in many machine learning (ML) applications. And such limitations arise frequently with Graphics Processing Units.

The limited memory size of common machines is one point that is often overlooked. The auxiliary data generated during the computation is typically extremely large and may exceed the size of the input data itself. Many (if not most) users of ML lack supercomputer-scale machinery, required for some ML algorithms. Virtual memory systems could be an option, but allowing the algorithm to run though the paging activities may hurt the overall performance.[3] For R-language software, the **bigmemory** package [73] allows the entire dataset to be accessed on disk, with minimal overhead (according to the authors.) Neither of these is an entirely acceptable option, and in the case of GPUs, these also have limited apace and availability.

These considerations naturally lead to chunked approaches on clusters of CPUs or GPUs. Since each chunk runs on a different machine, and the chunks have a smaller memory footprint, memory limitations may be overcome.

[1] The time complexity of an algorithm is described using *Big-O Notation*, when n is the size of the input and 0 is the growth rate function for the worst case scenario for run time.

[2] The amount of speedup is always in the denominator of the equation, meaning the complexity is a factor of that much faster.

[3] In the case of a shared machine, a user may not even be able to increase swap space size

1.2.6 Potential for SA in the GPU Realm

There is currently a keen interest in multi-GPU systems [115]. SA should have excellent potential here, in two senses:

- In a multi-GPU setting, chunking is a natural solution, hence SA.
- A perennial problem with GPU applications is the limited amount of memory available. This too makes SA a natural choice, even if the chunks are fed into the GPU serially, just to be able to execute the algorithm. But there is more:

  Recall the rough time complexity analysis of Section 1.2.4. If the chunks must be handled serially, that implies a total time of roughly $O(rn^d/r^d) = O(n^d/r^{d-1})$. Thus one can still attain a speedup if $r > 1$, and in fact a superlinear speedup if $r > 2$. So, not only does SA provide a convenient way to deal with GPU memory limitations, it may also produce a speedup even in the single-GPU case.

Let's consider this in more detail.

1.2.7 Deep Learning

The CPU is not designed to handle deep learning or the huge databases often used in the era of Big Data, so implementation designed for GPUs is important area of research. For example, the standard GPU (e.g. NVIDIA GM206) does not have enough memory to store the 10M movieLens data. Thus, Software Alchemy on multiple GPUs, or even serial SA on a single GPU, gives us a convenient solution.

The popular use of neural networks for image classification is an example in which GPUs can be beneficial. For instance, training a the network model (such as GoogleNet, AlexNet, or ImageNet) could take days or even weeks on the CPU or single GPU [132].

Currently the only solution to train models larger than one's GPU is to use multiple GPUs, reduce the model size, or reduce the batch size [132]. Again, SA would seem to be a better approach.

1.3 Specific Aims

A number of previous authors have looked at ways to parallelize specific ML algorithms, such as [25] [159]. However, *the key potential advantage of SA is its generality*. The aim with SA is to provide users with a universal tool to parallelize any kind of ML algorithm. While this may be a rather ambitious goal, **the purpose of this study is to demonstrate that SA can in fact be an effective tool for parallel computation of many popular ML algorithms**.

1.4 Definitions and Background

1.4.1 Recommender Systems

Recommender systems (RS) are of particular interest here, and there will be a focus on a subfield known as *collaborative filtering*, which combines user and item ratings data (as opposed to, for instance, inferring user ratings from textual input).[4] For this, the feasibility of SA will be explored in this context using an R package called **rectools** designed inspired by that field [120] [105].

Recommender systems are best known in their usage for predicting consumer product ratings. The "Hello World" example for recommender systems is the MovieLens data [55]. Here the data are in a standard format, consisting of (userID, movieID, rating) triples. Most users have not rated most movies, and the goal is to complete that ratings matrix. This process is often called the *matrix completion* problem [19].

A rating might be on a numeric scale, such the 1-5 range in the MovieLens data, or simply be binary, e.g. the Like button in Facebook and Twitter.

There are many nonmarketing applications of RSs as well. For example, in some applications of random graphs one is interested in predicting where new links might arise [4]. Here both the "users" and "items" are vertices in the graph.

1.4.2 The Number of Chunks

There is also a question of how many chunks to use. The asymptotic proof in [107] keeps the number of chunks r fixed while the number of data points n goes to infinity. On the other hand, it is found that in practice there is a "U-curve" in some situations, where the accuracy gets lower with increases in chunk number

Therefore, experiments will be limited to a small number of chunks, (typically two, four, and eight), on a set of standard quad-core machines with hyper-threading factor two.[5] In most cases we stay on a single machine, but since the machines are in a cluster, we also present some results in the cluster context.

In each case, we list both run time (criterion (a)) and accuracy (criterion (b)), either mean absolute prediction error or proportion of correct classification.

1.4.3 The Impact of Tuning Parameters

It has been demonstrated that SA works quite well in the case of parametric models; see for instance [107] and the references cited there, and [2]. Note; that "works quite well" means that

(a) a good speedup is obtained, *and*

[4] Various definitions differ on this.

[5] Most of the experiments were performed Intel Core i7-4790K machines, running at 4Ghz. A few were done on a similar laptop machine.

(b) comparable accuracy is maintained.

In the prediction context, for instance, criterion (b) means that SA achieves the same level of, say, mean absolute prediction error. The fact that (b) holds for parametric models is proven in [107].

However, it is not clear *a priori* that this success can be extended to typical machine learning (ML) methods. A major issue concerns tuning parameters (also called *hyperparameters*). Consider the simple example of k-Nearest Neighbor methods. For any given number of data points n, there is an (unfortunately unknown) optimal number of nearest neighbors. If we use SA with, say, eight chunks, then the optimal number of nearest neighbors k will be different for the full data set at the chunk level.

Many ML algorithms make use of several tuning parameters. One interesting example is that of SIS (Sure Independent Screening), a Lasso-class method for choosing features in ultrahigh-dimensional space [43], which has so many tuning parameters (eight) that the paper displays them in a table. The choice of values for the tuning parameter(s) is typically chosen via cross validation. That of course can be done at the chunked level as well, but subtle interactions may occur. Thus, the extension of SA to the ML realm, though intuitively promising, must be confirmed empirically.

1.4.4 "LEAVE IT THERE PHILOSOPHY"

The **partools** package has a *Leave It There* theme. This means keeping a dataset distributed among computational nodes for the duration of a session, i.e. through the many different analyses one might run, so that data transfer costs are amortized. Permanent data thus involves distributed files, in the spirit of Spark/Hadoop; **partools** has provisions to facilitate this (though more explicit and less elaborate than Spark/Hadoop).

For example, consider this typical scenario in the use of R's **parallel** package, which does a scatter/gather operation. Say we wish to perform the following on some dataset:

- Convert categorical variables to indicator variables.
- Replace NA (missing) values by means.[6]
- Remove outliers, as defined by $|X - \mu| > 3\sigma$.[7]
- Run linear regression analysis.

The point is that, after the manager node does a scatter operation to distribute the data to the worker nodes, we should NOT do a gather operation back to the manager at the end of each of the above steps. A gather operation is postponed as long as possible, in order to avoid the communication overhead;[8] distributed operations are used as much as possible.

[6] Not necessarily a good idea, just an example.

[7] Again, just an example.

[8] R's **parallel** package uses direct sockets by default but offers MPI as an option. In practice there usually is not much difference in performance.

The **hyperquicksort** function, written by the author, can be used to speed up the sorting operation, while keeping data distributed to facilitate this. This algorithm can be done while keeping the data distributed in order to avoid the communication overhead that is often created since most sorting algorithms require all the data to be on the master node. The function works by picking a 'pivot point' the data in each chunk (on each CPU/GPU node in the cluster), and sending the data above the pivot to the node above and the data lower than the pivot to the node below. This is done until each node in the cluster has all the data that is closest in value (eg. all the lowest, highest, or middle if there are three nodes). Then, finally, the subsets of data on each individual chunk is sorted so that all the data as a whole is also sorted when in is brought back to the master (if needed).

Note that even the regression operation can be done in distributed fashion, using SA [107]. Moreover, hopefully the data would be in a distributed file in the first place. Thus there would be no interprocess communication overhead even at the data input stage.

This is an extremely simple idea, but like SA is very powerful. It will turn out to play a key role in the efficacy of SA below.

1.4.5 Maximum Likelihood Prediction Model

As noted, the time and accuracy efficiency of SA is well-established for parametric models. So, they will be explored here as a basis for comparison.

Model The model is a somewhat more statistical treatment of what is termed a *baseline* model in the RS literature. Following [48], set

$$Y_{ij} = \mu + \gamma' X_i + \alpha_i + \beta_j + \varepsilon_{ij} \tag{1.1}$$

where

- Y_{ij} denotes the rating of item j by user i. Most users have not rated most items, so in most cases Y_{ij} is unknown.
- μ is a fixed but unknown overall mean.
- X_i is an optional known vector of covariates (e.g. age, gender) for user i, and γ is a vector of fixed but unknown linear regression coefficients.
- α_i is a latent effect for user i.
- β_j is a latent effect for item j.
- ε_{ij} is an error term.

The α, β and ε quantities are assumed independent Gaussians with mean zero and unknown variances. A Maximum Likelihood approach is then used to estimate them, and thus predict the ratings for the unknown user/item combinations.[9]

As pointed out in [48], for large data sets, the computational cost in time and space can be quite substantial, on the order of $O((R+C)^3)$ time and $O(R^2+C^2)$ space for a Y matrix with R rows and C columns, i.e. R users and C items. Hence there may be a strong potential value to SA here.

1.4.6 Nonnegative Matrix Factorization

The general notion of nonnegative matrix factorization (NMF) is as follows [80]. Given a very large matrix A, we wish to find low-rank, nonnegative matrices W and H such that

$$A \approx WH \tag{1.2}$$

In general ML applications, such as image or text classification, the entire matrix A is known. However, in the recommender systems case, the matrix consists of ratings, say with users being rows and items being columns, and it is mostly unknown. After finding W and H, one finds their product and thus has predictions for all elements of A, i.e. for all item ratings for all users.

1.4.7 A Modified SA

In applying NMF to recommender systems, we have each process find W and H for its chunk, as usual for SA. But one cannot simply have the manager average the values of W and H, as in standard SA, as the factorization (1.2) is not unique for a given chunk. The situation is similar to that of Singular Value Decomposition (SVD),

$$A \approx U\Sigma V' \tag{1.3}$$

If we multiply one (U,V') pair by -1, we still have an equally valid decomposition. So if one chunk has a "positive" version and another negative, averaging the U values over all chunks makes no sense. And though one might average the values of the product WH across the chunks, the product may be quite large.

Thus we form the analyst's desired predictions separately by each chunk, then apply averaging to the predicted values across chunks. Note that this is similar to the notion of V-SA, introduced in Section. 1.2.3.

[9] These combinations are only for users and items already in the dataset. If, say, we have a new user, the *cold start* problem, we can at least compute $\mu+\gamma' X$ if the value of X is known for that user.

1.4.8 k-Nearest Neighbor Model

Consider user A, who may or may not already be in our data, and for whom we wish to predict a rating for item B. We do so by first restricting our data to those who have rated B, and then compute a "distance" (or "similarity") from A to each of these users. The distance is computed as a cosine between the ratings of A and those of the other users, on items rated in common: The dot product is formed between the ratings vectors of B and the other user, divided by the product of the norms of the two vectors. This is probably the oldest of collaborative filtering methods, and still widely used.

1.4.9 Recommender System Prediction As a Regression Problem

One of the functions in our **rectools** package, **regressYdots()**, treats the matter as a regression problem, with user means and item means as predictor variables. Each user is given a feature value equal to the mean among that user's ratings over all items. Similarly each item is associated with an overall mean.

We can then predict **rating** from **userMean** and **itemMean**, as well as any covariates for the user and/or item. The prediction can be done by any method, whether parametric or ML, say neural nets. Performance (not shown here) has been shown comparable to that reported for general ML. Further investigation is needed.

1.4.10 SA in General ML Applications

Though we have focused on recommender systems here, SA should be a highly useful speedup tool in general ML applications. In this work, the notion on a few data sets and algorithms is explored.

Again, we hope that

(a) a good speedup is obtained, *and*

(b) comparable predictive accuracy is maintained.

Concerning (b), it will be seen that SA may even bring some improvement in predictive accuracy, presumably due to smoothing effects SA attains in the often-volatile ML algorithms.

1.4.11 Comparison to the MapReduce Programming Model

The MapReduce programming model is used for processing and generating large data sets with a parallel algorithm on a cluster. This paradigm was designed to meet the need for efficient computation in the era's increasingly common distributed systems. Basically, the method component filters and sorts the data, while the reduce phase processes and summarizes the output. The specific programming framework being used manages data transfers between parallel servers and performs the distributed shuffle operation, where optimal configurations minimize the cost of this communication. Multithreaded applications can achieve fault tolerance by taking advantage of the optimized system-wide sort operation, but single threaded applications will not benefit from this model.

MapReduce usually goes through three main steps on a distributed filesystem or database. These consist of a map-phase, a shuffle/sort phase, and a reduce phase [104]. In the map phase an associated key-value pair is associated with the input data. In the shuffle/sort phase, the output of each mapper having the same key are combined on each worker node, while in the reduce phase each parallel reducer then process data in parallel and sorts all of the output having a specific key.

For a very simple example, say we have a dataset consisting of working hours for each day of the year and we want to find the day of the week with the most hours (then we have multiple times for each day Monday through Sunday). Each day is the *key* and each number of hours is the *value*, and the mapper function that is handling each separate set counts its own data. Between the map and reduce phases, the data is normally shuffled to move it from the map node that produced it to the shared one where it is reduced. If there are only two mappers then each one would get the total for each day for their own half of the set (of all the files), then the reduce function would be used to sum both the totals (from each mapper) for each day. Our result on the master node would then be the total sum from each node, where we could apply the max function.

There is usually a straight-forward way to apply the MapReduce programming model (given the constraints). Linear regression would typically be implemented to compare the programming model to SA. We know that the estimate of $\hat{\beta}$ is obtained from $X^T X \cdot \hat{\beta} = X^T y$, where X is our matrix of predictor variables and y is the vector of predicted values. Our first mapper function implements $X^T X$ and a second implements $X^T y$. We then break our matrix X and vector y into sub-matrices X_i on each cluster node, so:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{bmatrix} \tag{1.4}$$

The mapper functions are applied to each sub-matrix, and then a reduce function is used to sum each separate matrix or vector element in parallel from each node:

$$X^T X = [X_1^T \;\; X_2^T \;\; X_3^T \;\; \dots \;\; X_n^T] \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{bmatrix} \implies X^T X = [X_1^T X_1 \; + \; X_2^T X_2 \; + \; X_3^T X_3 \;\; \dots \; + X_n^T X_n] \tag{1.5}$$

$$X^T y = [X_1^T \;\; X_2^T \;\; X_3^T \;\; \dots \;\; X_n^T] \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \implies X^T y = [X_1^T y_1 \; + \; X_2^T y_2 \; + \; X_3^T y_3 \;\; \dots \; + X_n^T y_n] \tag{1.6}$$

With SA, we solve for the linear regression coefficients on each separate node and then send our test data we would like to predict to each of them. The estimate is made based on the average of the predicted results. Using MapReduce, we send the average (reduced) results of each matrix multiply ($X^T X$ and $X^T y$) and solve for the coefficients on the master node. This comparison is similar to the section where we compare weights averaging versus voting since in the voting setting the prediction is determined based on the most chosen value among the models produced on each node. The only difference is that in linear regression we predict an integer value and not a class.

1.4.12 Comparison to the Federated Learning Programming Model

Federated learning is often used to solve the issue of the constant back-and-forth travel of data from CPU to GPU (eg. on another server, such as the cloud) in traditional machine learning [87]. Instead of one central device, models are trained locally on the host device before parameters (eg. weights) are averaged [57]. Unfortunately, this also requires frequent communication between nodes during the learning process and each node must have unrestricted access to the other nodes. A high amount of computation, memory, and communication bandwidth are needed to send constant updates on model information such as gradients [87]. This also could lead to reveal of sensitive information since each server may posses a different organization's dataset.

On the other hand, SA does *not* exchange the actual parameters of the ML model to generate a shared model (and there is no shared model). Each node has a separate/unique model which is used to evaluate the results on only its own data, before averaging the final predicted values. In SA data is assumed to be i.i.d., and comes from one dataset.

In federated learning, dataset may vary significantly in size, type, and value since it may originate from different organizations, devices, etc. [57].

1.4.13 Theoretical Foundations

Software Alchemy is fully efficient if the sample parameter $\hat{\theta}$ can be shown to be asymptotically normally distributed. If the data are i.i.d. vectors $X_1, X_2, ...X_n$, then the chunked estimator $\tilde{\theta}$ will have the same asymptotic variance as the full estimator. (See Section 3 of [107] for the full derivation of this, using characteristic functions.)

Let us address this and related aspects for the work here.

Maximum Likelihood Estimation:

It can be shown that under fairly broad assumptions, MLEs are asymptotically normally distributed [44]. Thus MLEs are covered by SA.

Principal Component Analysis:

The **recosystem** package gives the user a choice of NMF or Singular Value Decomposition, essentially Principal Component Analysis.

PCA can be shown to be asymptotically normally distributed by the following argument. Say we have a sample of random vectors X_i. The sample covariance matrix is

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} X_i {X_i}^T - \overline{X}\overline{X}^T \tag{1.7}$$

where $\overline{X}$ is the vector sample mean. Since we are working with sums, the Central Limit Theorem (vector version) tells us that $\overline{X}$ is asymptotically normal, and so is the first term in (1.7). By the Delta Method [33], (1.7) and functions of it, e.g. its eigenvectors, are asymptotically normal as well, and thus again covered by SA.

k-NN:

Asymptotic normality of k-Nearest Neighbor estimates is established in [100], so this too is covered by SA.

V-SA:

One can establish statistical consistency (convergence to true value as sample size goes to infinity) along the following lines.

Consider prediction at a particular point t, in a classification problem with (for convergence) two classes. Assuming a policy of guessing Class 1 when the estimated conditional probability is larger than 0.5, the key point is that as $n \to \infty$, the estimated probability will eventually rise above 0.5 and stay above it,[10] due to the strong consistency of the regression estimation process.

[10] With probability 1.0, i.e. pointwise convergence.

Non-negative Matrix Factorization:

Much theoretical work has been done on SA-like methods for the matrix completion problem. The interested reader is referred to [101].

1.4.14 When SA Can be Used

One important thing to recognize is that SA cannot be applied to all algorithms. For example, if the algorithm cannot be properly applied to a chunk of the data (eg. because it needs all the data points to evaluate the result), then it wont work. Some examples are certain image processing, sorting, or graph algorithms where the data needs to be kept in a certain order or structure to be useful. Reasons for these are listed below.

- Sorting: In most sorting algorithms we need to have all the data points visible at some step, to find the minimum, maximum or midpoint. Otherwise, the resultant chunks will be sorted, but not the data set as a whole.
- Graph Algorithms: In graph search algorithms, we need all the graph nodes points) available to find the shortest path because we don't know how the sub graphs are connected between each chunk. We could be left with even larger problems to solve after the cluster graphs are evaluated if there are more connections to data points in other chunks. The issue would be even worse if we did not give chunks connected sub-graphs, but this dividing up of the graph data also takes time.
- Image Processing: For example, if we have an image and we divide it into four equal chunks one chunk would not know what is in the other chunk. If we are doing object detection, only one chunk would have the object within the frame, so three out of four of the chunks would probably predict no-object (which is wrong).

1.5 Evaluation Methods

1.5.1 Accuracy Calculation

The mean absolute prediction error is calculated by summing the predicted values subtracted from the actual values and dividing by the number of samples. This is shown below where e_t is the error for the t number sample. The proportion of correct classification is the number of correctly predicted samples divided by the total number of samples. This is shown below where p_t is only 1 if the correct class is predicted with the highest probability.

Mean absolute prediction error	$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Proportion of Correct Classification	$\text{PCC} = \frac{1}{n}\sum_{t=1}^{n} p_t$

1.5.2 Datasets & Preparation

In this paper, I present a sampling of our results, involving the following data sets:

- **MovieLens Data**

  This data was mentioned earlier. It comes in various sizes. Here the versions with approximately one million and 20 million input records were used.

- **Book Crossings Data**

  This data, maintained at the same site as MovieLens, concerns book reviews. There are approximately one million records.

- **Jester Data**

  This data, maintained at the same site as MovieLens, concerns joke reviews. This data includes about six million ratings.

- **New York City Taxi Data**

  The dataset is from Kaggle.[11] It consists of records of over one million taxi trips, with records of trip time, vendor, number of passengers, and latitude and longitude for pick-up and drop-off. We predicted trip duration, either directly or the event that duration exceeds 450 seconds.

- **Last.fm Data**

  This data set[12] involves predicting popularity of songs.

- **Forest Cover Data**

[11] https://www.kaggle.com/c/nyc-taxi-trip-duration

[12] https://labrosa.ee.columbia.edu/millionsong/lastfm

This is a famous data set from the UCI Machine Learning Dataset Repository.[13] One attempts to predict one of seven classes, which are forest cover types, from variables such as Hillside Shade at Noon. There are about 570,000 cases.

- **Large Movie Review Data**

  The Large Movie Review Dataset (often referred to as the IMDB dataset) contains 25,000 binary movie reviews (either rated good or bad) each in the test and train sets [99] . Using this dataset we attempt to determine whether a given movie review has a positive or negative sentiment. Neural networks are often used to make predictions on this data set since words are encoded as real-valued vectors in a high-dimensional space.

- **CIFAR Data**

  The CIFAR-10 data is a labeled subset of the 80 million tiny images dataset. It is used for image classification, and consists of 60,000 images of 10 classes. In total, there are 50,000 training images and 10,000 test images [78].

- **CASIA Image Data**

  CASIA is a color image database with realistic tampering operations, made publicly available for researchers to compare and evaluate proposed tampering detection techniques [38].

- **prgeng Data**

  This is data from the 2000 US census, consisting of programmers and engineers in Silicon Valley, about 20,000 cases.

Though we cite numbers of records above for these datasets, it should be noted that this measure may tell only part of the story. For instance, the (smaller) MovieLens dataset and the Book Crossings dataset each have about one million records. Yet the books data has over 100,000 users and 340,000 books, which dwarfs the corresponding values of about 6,000 and 4,000 for MovieLens. Any given RS algorithm thus has much more work to do for the books data.

Our aim in this paper is to illustrate the use of SA on several ML algorithms, first for recommender systems and then for general ML contexts, on various real data sets. Note that no attempt is made here to choose "good" values for tuning parameters. the fact that some methods have greater predictive power than others is tangential to our goal, which is simply to speed up whatever method a user chooses.

Similarly, no data cleaning was performed. The taxi data in particular is outlier-prone.

1.5.3 Software & Hardware

The author was a coauthor of the following R packages used in the following experiments.

[13] https://archive.ics.uci.edu/ml/datasets/covertype

R PACKAGES

- The **rectools** package[14] offers tools used for predicting users' ratings in recommender systems. It includes features such as parallel computation, automated cross validation, and incorporation of covariates, say demographic information about the users [15] [120].
- The **partools** package [106] aids in preparing the environment for parallel and distributed computation. After using **setclsinfo** to give each node a unique ID, **distribsplit** splits the full data set into equal sized chunks and sends them to each of the worker nodes, with the option of being randomly selected or not.
  - The author's **hyperquicksort function** is available on GitHub in **partools**, and includes test cases and instructions for public use. **Hqs.R**: https://github.com/matloff/partools/blob/master/R/hqs.R.
- The R package **recosystem** was used to perform the NMF operations. The package is fast, using C++ as its core, and making use of OpenMP. One would hope that superlinearity would give a further performance gain, on top of OpenMP.

Unless otherwise stated, default values are used for optional arguments, such as reduced rank in matrix factorization.

THE USE OF SA WITH THREADED ALGORITHMS SA is typically used with stock, off-the-shelf algorithms and software. For example, in our experiments with Nonnegative Matrix Factorization below, the R **recosystem** package [119] was used. In our Maximum Likelihood Estimation experiments, another R package, **lme4** [10] is used. In both cases, the given package is applied to each chunk, then combine the results.

Both of these packages are written and very finely tuned by prominent researchers. In particular, the core code in both packages is written in C++ using OpenMP. Say one has a quadcore machine. (Ignore possible effects of hyperthreading.) Then with the full data, i.e. not using SA, we are already fully utilizing the machine. Yet, if the algorithm in the package is such that SA could yield a super-linear speedup, SA may still produce a performance gain.

GPU RESOURCES The tests with CUDA Streams **??** was tested on an NVIDIA GM206 GPU with a fast GPU version of the kNN algorithm [50].

In 1.6.11 the CNN was designed with hyper-parameters from a popular blog [164], applying independent chunks of the training data to the Quadro 4000 GPU and averaging each of the GPU results.

[14] http://github.com/matloff

[15] The term *covariate* for example, is statistical. In the RS literature, the term *side information* is common.

chunks	scatter	train.	pred.	mean abs. error
full	-	1114	0.46	2.67
2	5	686	0.5	2.72
4	11	423	1	2.77
8	11	247	1.5	2.82

Table 1: MLE Model, Book Crossings Data

chunks	scatter	train.	pred.	mean abs. error
full	-	99	0.3	0.710
2	5	100	0.3	0.737
4	3	73	0.5	0.752
8	8	100	0.5	0.764

Table 2: MLE Model, MovieLens Data, 1M

1.6 RESULTS

The author carried out each of the experiments below and reported the following data: data scatter time, model training time, and prediction time each in seconds. In each table below, The accuracy is either the mean absolute prediction error in a regression problem or proportion of correct classification in the case of classification.

1.6.1 MLE

The results for the Book Crossings data were as in Table 1. Here I used Location and Age as covariates, for prediction of 10,000 new cases. Note that under the Leave It There and distributed files philosophy of **partools**, data scatter time would be 0.

The results for the Book Crossings data were as in Table 1. There is an increase in prediction time, due not being able to apply the Leave It There principle to prediction; SA requires a gather operation here in order to compute overall predictions at the manager node. Here the scatter time does not play a major role.

There is a nearly-linear speedup. We have found that the very minor loss of accuracy this can be ameliorated by randomly permuting the rows of the input dataset.

The results for the MovieLens data, size one million, were similar, as seen in Table 2. Note that with 8 chunks, we have already passed the minimum point of the “U-curve” of run time versus number of processes that is typical in the world of parallel computation.

As noted, the base software used in the MLE context here, **lme4**, is highly threaded. Thus the above results

chunks	tot. time
2	248
4	1568
8	191

Table 3: MLE Model, Book Crossings, Cluster

# of chunks	time	mean abs. error
full	87	456.00
2	48	451.08
4	26	392.13
8	17	424.36

Table 4: k-NN, NYC TaxiData

indicate a super-linearity effect. To explore that further, we also ran on a cluster of 4 machines.

Recall that the time for the full data was 1114.155 + 0.455 = 1114.61 seconds. We are seeing a pronounced super-linear effect, e.g. a speedup of 4.50 for only two chunks.

1.6.2 A Modified SA

1.6.3 k-Nearest Neighbor Model

This was tested on the NYC Taxi data, using $k = 25$. The results, using the **regtools** package [103], are reported in Table 4. Again a similar pattern — substantial speedup, with a bonus of somewhat improved predictive accuracy.

The results on the Jester data, with $k = 25$, are shown in Table 5. Again, SA achieves superlinear speedup, at least for two or four chunks.

SA also brings improved prediction accuracy, which as noted in Section 1.4.3 shoud be viewed with caution. Actually, the results suggest that the value $k = 25$ may be too low for use with the full data set. Again, the issue of setting tuning parameters at a different level for the chunks than for the full set warrants further study.

1.6.4 Logistic Regression

Let us begin with a parametric model common in both statistics and ML, logistic regression. One of our experiments involved predicting trip time in the New York City taxi data.

I used logistic regression to predict whether a trip duration would exceed 450 seconds. For this particular exper-

# of chunks	time (sec)	mean abs. error
full	260	4.79
2	76	4.60
4	58	4.36
8	81	3.89

Table 5: k-NN Model, Jester Data

# of chunks	time	prop. correct class.
full	41	0.694
2	39	0.694
4	24	0.694
8	14	0.694

Table 6: Logistic Model, NYC Taxi Data

iment, I ran the algorithm on a machine with a larger number of cores (though with slower cores). Our goal in that was to investigation how much chunking the setting could tolerate. The machine had 16 cores and a hyper-threading degree of 2.

The results were as in Table 6. It was found that the run time was decreasing even at 32 processes, though the overall times do suggest the ”U-shaped” runtimes common in parallel computation was imminent. Note that the chunked and full versions all had the same prediction accuracy, 0.694.

1.6.5 Neural Networks

A standard R package, **neuralnet** [46] was used at first. There are of course many packages for this methodology, and a virtually unlimited number of tuning parameters for each, no attempts for such optimization were made in these experiments. The Last.fm data was tested with 5 nodes in 1 hidden layer. Results are reported in Table 7. A nice speedup is obtained, and possibly even an improvement in prediction accuracy.

1.6.6 Convolutional Neural Networks (CNN)

The use of SA was testedon the IDMB data using the R **keras** package which is written as an interface to the Python package Tensorflow. The test was based on an example from the Tensorflow for R webpage examples [1]; refer there for the hyperparameters used. SA reduced computational time with clusters of 2 and 4 but the U-curve begins to take effect as the cluster size increases to 8.

# of chunks	time	mean abs. error
full	486	221.41
2	326	211.94
4	254	210.15
8	133	221.41

Table 7: Neural Nets, Last.fm Data

# of chunks	time	prop. correct class.
full	191	0.86
2	160	0.86
4	177	0.87
8	241	0.83

Table 8: CNN, IDMB Data

1.6.7 Random Forests

It is only fitting to try a random forests approach to analyzing forests! The R package **randomForest** [89] was used on the Forest Cover data, with all 55 features in the dataset. See Table 9 for the outcome. Speedup is approaching linear, but with comparable predictive accuracy only for the 2-process case. This may be due to the extreme imbalance in class sizes, e.g. 283,301 records for Cover Type 2 but only 2747 for Cover Type 4; some chunks probably had very few of the later.

1.6.8 Support Vector Machines (SVMs)

The SVM algorithm was testedon the CASIA image data [38] using R's **e1071** package. In this test predictions were based on the RLRN (Run Length Run Number) feature vectors of length 60 (15 run lengths in each of the 4 directions)

# of chunks	time	prob. correct class.
full	842	0.955
2	485	0.941
4	237	0.919
6	195	0.911

Table 9: Random Forests, Forest Cover Data

# of chunks	time	prob. correct class.
full	649	0.97
2	2189	0.94
4	919	0.94
8	599	0.93

Table 10: SVM, CASIA Image Dataset

exactly as in [110], to make the binary classification of an image as being either tampered or authentic. SA produces a super-linear speedup factor of almost 3 for a cluster of 2, and still maintains adequate prediction accuracy.

1.6.9 Missing-Value Imputation

The presence of missing data is often a vexing problem in ML applications [140]. There is a vast literature on missing-value (MV) methods, and a corresponding array of R packages. Here we used the **mice** package [160].[16] As the package tends to do voluminous computation, it is a good candidate for SA.

The experiment was designed as follows: 10% MVs were artificially injected into the data (coded NA in the R language), and performed linear regression analysis, predicting Wage Income from Age, Gender and Weeks Worked. Ten runs were done, each using **mice** both with SA (two chunks) and with the full, i.e. unchunked data.

Speedup with definitely superlinear. A typical pair of runtimes was 96.82 seconds for SA, 369.07 seconds for the full data — a speedup of 3.81 for only 2 chunks.

Furthermore, SA was as accurate as the full method. Here are the mean estimated regression coefficients over ten runs, compared to the true values (no NAs):

method	Age	Gender	Weeks Worked
SA	4349	9952	1115
full	413	10117	1091
true	497	10701	1373

Clearly the MVs are costing us, but if **mice** is used as our remedy, SA will do as well as unchunked in accuracy, but much faster.

[16] Parameters were **m = 4**, **maxit = 50**, **method = ’pmm’**.

chunks	scatter	train.	pred.	mean abs. error
full	-	34	0.3	0.649
2	13	19	0.6	0.647
4	22	10	1	0.656

Table 11: NMF Model, MovieLens Data, 20M

# of GPUs	time	prop. correct class.
1	852	0.78
2	506	0.77
4	192	0.73
8	152	0.66

Table 12: CNN, CIFAR10 Data

1.6.10 NMF

In applying NMF, we have each process find W and H separately, and then apply averaging to the predicted values across chunks (similar to the notion of Voting-SA.) SA produced a somewhat sub-linear speedup on the 20 million-row Movie-Lens data. The speedup depends on using the Leave It There principle; if the data must be distributed first, SA would not be very helpful. Even better results would be produced by leaving the data scattered rather than redistributing it for the algorithm. As shown below, in the first column, a proportion of the runtime is spent on the scatter operation. However, there is a slight much deterioration in accuracy.

1.6.11 Deep Learning

A CNN model on CIFAR-10 was tested by chunking the data, sending it to a separate GPU, then averaging each of the GPU results. Table 12 shows that there is a great amount of potential for Software Alchemy on a multi-GPU system since the accuracy is mostly maintained while an approximately linear speedup is attained. Using 8 GPUs versus using only one reduces the runtime from over 14 minutes to only about 2.5 minutes. Essentially full accuracy is maintained for two chunks, but there is some sacrifice with larger SA configurations.

One important thing to note is the time spent transferring the data from the CPU to the GPU, and from GPU to CPU. Although GPUs are being increasingly used in general purpose computation, potential benefits could be offset due to latency of transferring between GPU and CPU.

This latency effect may differ depending on whether the GPU is on the cloud or not, what type of GPU is being

# of streams	reference points	runtime
0	32768	0.41
2	32768	0.20
0	16384	0.20
2	16384	0.10

Table 13: k-NN CUDA

used, how much data, and how often the data is transferred. As discussed in 1.4.4, in our attempt to keep the data scattered we likely only need to do this ”scatter” operation once in each direction for given algorithm. However, this time may be longer to send the data to or from a GPU than a CPU or another CPU core. It may be even longer on a cloud GPU due to the latency required to send the information to the remote CPU.

For example, the peak bandwidth between the device memory and the GPU differs depending on the type/model of GPU. On the NVIDIA Tesla C2050, for example, than the peak bandwidth between host memory and device memory is 144 GB/s while on the PCIe x16 Gen2 it is 8 GB/s [56]. Typical GPUs have CPU to GPU memory transfer speed of 5-10GB/s [47]. Therefore, it is something to take into account when setting up your program.

On the other hand, many algorithms (such as deep neural networks) are still a lot faster than if they would have run on the CPU, so it also depends on the algorithm itself).

1.6.12 CUDA GPU Streams

CUDA streams [58] are another possible approach to reduce computational time by applying this chunk averaging technique in the GPU framework. Streams can achieve concurrency by running multiple instances of a single kernel or separate kernels.

When the data is first broken into chunks (like in the CPU version), and run on different sets of blocks, the runtime of the algorithm also goes down since the number of data points to process per stream is smaller. Although we may use the same total number of threads and blocks there is less data to compute all at once. If multiple GPUs are not needed, this method saves time by not requiring the data to be sent between devices as all the data is gathered onto the same CPU from each separate kernel. On the other hand, if multiple GPUs are used and this method is applied on each node, we could potentially achieve superlinear speed up.

As shown in Table 13, an approximately linear speed up is obtained using multiple CUDA streams for each kernel of the algorithm (each using the same number of blocks, which is determined by the number of reference and query points).

Since the data was tested on randomly generated integers, it was more than likely that separate chunks would

# of chunks	W-SA prop. correct	V-SA prop. correct
2	0.88	0.88
4	0.87	0.87
8	0.83	0.83

Table 14: CNN, IMDB Data

each hold at least one of the nearest neighbors on our dataset. When the program was tested on small amounts of data, each chunk always held the same (correct) nearest neighbors (so these larger chunks tested also maintained 100 percent accuracy), but this may not be the case for other data sets, so the accuracy is dependent upon the variance of the data. Additionally,

Although this algorithm is quite fast enough on the GPU without chunking, it does solve the memory issue if chunked serially, as mentioned. Adding more data points in the tests above causes the full algorithm to fail due to insufficient memory.

1.6.13 Weights Averaging vs. Voting

M. Robins [133] reported using a technique amounting to SA in neural network (NN) analysis of very high-resolution medical images. The data was chunked, and NNs fitted to each chunk. Finally, the corresponding link weights were averaged across chunks to produce a new NN, which was used for prediction. This is similar to our logistic regression model in Section 1.6.4, but with NNs. This is contrast to our V-SA voting approach in Section 1.2.3.

When the effect of weights averaging (SA) was tested versus voting (V-SA) on the IMDB data with CNN, the accuracy was identical. For each size cluster 2, 4, and 8 we averaged the proportion of cluster nodes which predict the correct class over 3 trials as shown in Table 14 (where W-SA and V-SA refer to weights-averaging and voting versions of SA).

This is an interesting comparison, as voting may provide extra flexibility. For instance, one might perform different dropout operations to NN nodes in different chunks. In this and other ways, there is a possibility that the model produced by each node could be more tuned to its own chunk. If W-SA and V-SA continued to be equally accurate, the latter might be preferred.

1.6.14 Comparison to the MapReduce Programming Model

Table 15 shows a comparison of the percent accuracy of the prediction on one thousand test set ratings of (the 1 million subset of) the MovieLens data. The column labeled *1 Mapper* is the prediction accuracy for the MapReduce programming model where the cluster nodes each evaluated both of the matrices calculated above in one function. The tests

chunks	SA	1 Mapper	2 Mappers
2	0.93	0.93	0.94
4	0.92	0.94	0.94
8	0.93	0.94	0.93

Table 15: MovieLens 1M Percent Prediction Accuracy (MapReduce vs SA)

in the *2 mappers* column were done in the standard way MapReduce is used to implement linear regression, described above using multiple mapper functions [5] [49] [134]. The versions of MapReduce require making predictions on the master node, while in SA predictions are made on the cluster.

Interestingly, the SA method consistently had slightly higher accuracy, and this difference may be much more significant if the dataset were larger. Another benefit of the SA method is that it doesn't require figuring out how to parallelize the model since the entire model is built on each individual node. If we were working with a more complicated model, this could be even more of an issue. The reduce part of MapReduce model also requires moving data among nodes to combine the results from each node in the cluster, which could make the parallel computation use up unnecessary additional time, space, and resources.

Times are not listed since it was less than a second and neither method was faster than the other in any test. If all computation were done on a single node, with no parallelism of any kind, then the time complexity would be $O(np^2)$ for $X'X$, $O(np)$ for $X'Y$, and $O(p^3)$ or $O(p^2)$ for $(X'X)^{-1}$ (depending on the algorithm used), where p is the number of predictors and n is the number of samples. The final product can then be computed in $O(p)$ for $X'Y$ and $O(p^2)$, but neither MapReduce nor SA can do anything about the p factors; they can only replace n by $\frac{n}{r}$, where r is the number of network nodes (by separating rows among nodes). So, neither one of them can achieve dramatic speedup, and in principal both should have about the same speedup. On the other hand, on practice, MapReduce should be much slower, due to the ”shuffle” operation, which is a forced network-wide sort that is performed whether it is needed or not. In the case of this example, and in the case of most machine learning algorithms, it is not needed.

Hadoop is currently one of most popular MapReduce libraries and is desirable for its high level of fault tolerance. It is written in Java, and although it provides interfaces to other languages, it is easiest used in Java. One issue with Hadoop is that cannot keep intermediate results in memory between runs, but the Spark package is designed to remedy this. Hadoop and Spark are good options for the specific types of applications they are designed for, such as SQL, but have limited usability in the types of computation that common (non-computer expert) users need for their CPU-bound applications and large data set sizes. **Partools** is not only easier to setup, install and program for the everyday users, but the distributed file or object nature is retained. It also keeps data chunked to avoid network overhead, while the fault tolerance features of Hadoop and spark can often slow down computation.

A number of modifications to MapReduce have also been developed [81], but these are best for the types of problems are that are *Embarrassingly parallel (EP)* [28], since the reduce must involve little or no computation. Software Alchemy is an efficient method to convert non-EP into EP problems [107]. It is important to note that this does not mean either method is more accurate than the other or that there is an 'exact' value to compare to since each are estimations of the population.

1.7 Conclusions

SA is an easy, natural, *general* method for parallelizing ML applications. It was found it to be especially useful in recommender systems settings, typically with super-linear speedup, and often even with improved predictive accuracy, rather than just comparable accuracy.

1.7.1 Future Work

Investigation of the efficacy of SA in GPU contexts is a priority goal for future work. Another avenue to be pursued is our polynomial alternative to neural networks [24]. We have found the polynomial approach to work well on a variety of datasets, but it often has a large run time, which SA may be good at solving.

As discussed in the next section, SA can also be used as a tool to speed up the sliding-window method applied to images in detection algorithms.

2 Improving Image Fraud Detection Algorithms

2.1 Abstract

With technological advances leading to an increase in mechanisms of image tampering, our fraud detection methods must continue to be upgraded to match their sophistication. One problem with current methods is that they require prior knowledge of the method of forgery in order to determine which features to extract from the image to localize the region of interest. Here it will first be shown how certain classic image fraud detection techniques can be modified, combined or parallelized to produce improved results compared to the original, stand-alone or serial technique.

A machine learning algorithm can better pick up a large variety of features simultaneously and automatically by training on an entire database of full images. However, we are still left with the question of how to localize the manipulated region. To solve this, object detection networks such as Faster RCNN [130], which combine an RPN (Region Proposal Network) with a CNN may be adapted to fraud detection by utilizing their ability to propose bounding boxes for objects of interest to localize the tampering artifacts.

Therefore, second, it will be shown that a Multi-stream Faster RCNN network can be applied similarly, but with the second stream having an input of the ELA (Error Level Analysis) JPEG compression level mask. This method provides an even higher accuracy by adding training features from the *segmented image map* to the network.

2.2 Introduction & Significance

Images are often trusted as evidence or proof in fields such as journalism, forensic investigations, military intelligence, scientific research and publications, crime detection and legal proceedings, investigation of insurance claims, and medical imaging [109]. Furthermore, with the increase in digital data available sharing information in pictures and video recordings has increased, dramatically. The same useful image manipulation computer applications used to glamorize media or create realistic effects in fictional movies (eg. *Harry Potter* or *Twighlight*) for commercial purposes can be used in harmful ways as well.

It has been used to pass on convincing messages and propaganda. For example, most of the research done in digital crime investigation involves retrieving and investigating data that exists on machines [67]. This task can be extremely difficult for a human to collect, process, and analyze by hand, especially with large data and images. In order to protect legal and political photos while maintaining research integrity or authenticity, image manipulation detection is a very necessary tool [32].

Moreover, as technology advances, common image tampering techniques are widely available to the public. Worse yet, this often includes post-processing such as Gaussian smoothing, making it even more difficult for humans to recognize the tampered regions with the naked eye. Due to the difficulty of distinguishing fake and authentic images, research in this field has become integral to preventing hacking.

2.3 Related Work

Image forgery can be classified into two categories, active and passive, as shown in Figure 1. Active methods are used to embed the image with data, authenticate, or label it. These methods include digital signature and digital watermarking techniques, which are used for copyright protection and authentication respectively. Active methods require prior information about an image while passive methods do not. Passive methods include intrinsic regularities and inconsistencies, tampering operations, and natural and computer graphic image [155].

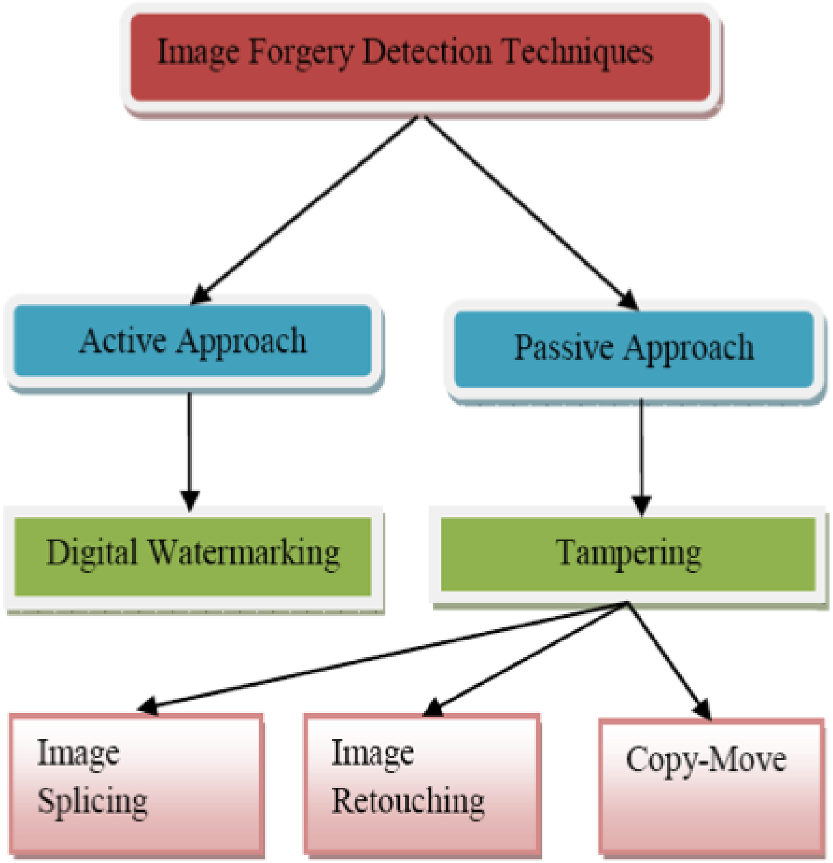


Figure 1: Active vs. Passive Image Forgery Techniques [146]

The main focus in research is on tampering operations which are mainly used by manipulators to deceive another party. Detection of passive techniques is a major challenge due to the various methods of detection that may be used.

Passive techniques include such as *retouching* or *resampling* which involves geometric transformations on part of the image, *image splicing* which involves pasting a part of one image to another image, *copy-move* fraud which involves copying one part of the same image to another location within the photo [74], or *removal* which is the elimination of a region of the image by in-painting [186]. Other techniques include median filtering, erase-fill, resizing, composting, seam curve, blending, matting, contrast enhancement, multiple JPEG compression, and color adjustment [155].

MANUALLY DESIGNED FEATURES Early image manipulation detection methods were originally designed to target a single type of tampering using simple feature extraction methods followed by classification [155]. Hand-crafted features are those characteristics of forgery in an image that must be described and well-known by hums/experts in the field. We must tell the detection software which image features to look for and then how to look for them. For example, the typical structure of a hand-crafted image manipulation feature extraction method is shown in Figure 2.

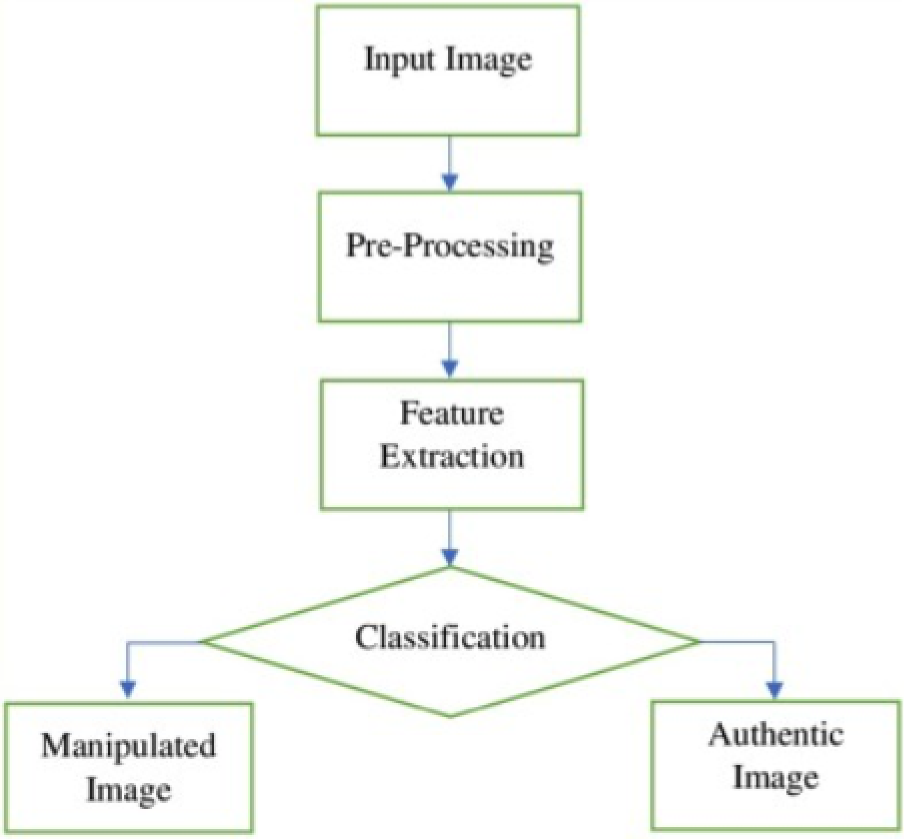


Figure 2: Typical Structure of a Hand-crafted Image Manipulation Feature Extraction Method [155]

CLASSIC DETECTION METHODS Some of the oldest and most common detection methods in *past* research include the color filter array (CFA)[20], image noise analysis using noise filters [94], Discrete Cosine Transform (DCT) frequency analysis [45], Discrete Wavelet Transform (DWT), JPEG compression measurement using methods such as Block Artifact Grid (BAG) [181], and ELA (Error Level Analysis) [54], local binary pattern (LBP) [34], scale-invariant feature transform (SIFT) [34], speeded up robust features (SURF [13], and many modifications of these [12] [27] [90] [184].

LIMITATIONS OF CLASSIC METHODS However, in real-world scenario, a single image can be altered by a large variety of image manipulation techniques, thus hand-crafting feature detection has nearly become useless in current real-world image fraud. Most classic image fraud detection algorithms is that they require prior knowledge of the method of forgery in order to determine which features to extract for localization.

For example, detection of forgery methods such as copy-paste fraud, added WGN (White Gaussian Noise), and color enhancements, each typically require different filters and algorithms (eg. PCA, DWT, or DCT-based) applied at different sized bounding boxes depending on the size of the tampered region [45]. Results may also depend on the image type, such as JPEG, PNG, or TIFF [181]. Since this information is often unavailable we don't know which to use. Although modern machine learning based methods have mainly taken over, some of classic methods are still used today.

Modern/ML-based Detection Methods But now, images are being manipulated using multiple tampering operations in order to make it have the characteristics of realistic image. With the advancement in the editing tools, the unique traces left behind to reveal the type of image tampering has gone, and it is very difficult to detect the technique and the region in the image. Now, a single digital image undergoes various manipulation methods and it is nearly impossible to detect using these classic feature extractor methods. Later on, some new approaches were developed to detect multiple-image manipulations in images but these methods were limited to some set combination.

Clearly, method of detection that is both quick to compute with the ability to be generalized to various differences between images (or even new types of tampering) is still in need.

With the increase in image data available and efficiency of modern GPUs to handle bigger problems, there has been a continuously increasing interest in the application of machine learning for image fraud detection. These modern ML algorithms begin to address the issues with classic techniques by automatically learning features from an entire image database. In most cases they are designed to be run on a GPU, so they also prove to be both significantly faster and more accurate versus the methods such as those described above.

These techniques train a model to estimate the probability of the images feature map (or sub-image feature blocks) being tampered [75]. Convolutional Neural Networks (CNN), in particular, are well suited for image tampering detection, and have been shown capable of detecting textures, noise, and resampling very efficiently [15] [23] [26] [62] [172].

Most new research studies in the field conclude that deep learning outperforms the traditional hand-crafted feature extraction methods because of its' ability to quickly and automatically extract a combination of complex/hierarchical features[185].

However they also mention that there is room for improvement. To illustrate, most of these new methods are basic CNN modifications with various filters added (eg. targeting a specific type of forgery), which still lack the ability to be generalized to a wide range of forgery types and the ability to localize the tampered region. Further, some research suggests that the use of CNN alone for forgery detection is less effective because CNN only learns semantics or the content of the image [3].

To solve this issue, an object detection network, such as Faster R-CNN, which combines an RPN (Region Proposal Network) with a CNN, can be adapted to fraud detection in order to localize any tampered regions within images [186]. By making use of the computational powers of today's GPUs this method is able to employ a deep architecture within each of the sub-image regions while simultaneously achieving a fast run-time. This ability to localize multiple detailed features helps it to outperform the top classic methods such as ELA (Error Level Analysis), NA (Noise Analysis), or CFA (Color Filter Array) [186].

2.4 Specific Aims

The first aim here is to determine whether image fraud detection algorithms can be parallelized in order to speed up the processing time required for classic imaging algorithms. Specifically, the sliding window operation will be of interest operation since it takes a lot of time to run an algorithm on an 8 *x* 8 box about *image heightx image width* times (eg. when the sliding window shift is of length 1).

Second, classic methods will be developed to test if they can be improved enough to be comparable in efficiency to modern ML methods. Based on past work, it seems this could be possible by improving the speed via parallelizing and improving the ability to simultaneously detect multiple types of manipulation via combining algorithms. Specifically, in this work, multiple detection algorithms will be *combined* to develop a more robust system targeting multiple feature types. Then they will be parallelized to improve the runtime.

The third is to compare this to modern ML algorithms by testing on similar data. If ML algorithms prove to be more efficient, they will be optimized by supplementing the model with any missing features from classic methods. Software Alchemy will also be tested as a method to improve the speed.

The fourth goal is to test object detection methods such as Faster-RCNN and compare them to classic methods alone as well as any improved standard (non-deep learning) machine learning methods. Specifically, a Multi-stream Faster-RCNN will be used to combine classic features with the ability to localize specific regions in the image quickly and efficiently on a GPU.

2.5 Definitions & Background

2.5.1 Classic Methods

DCT & PCA Two of the oldest and most common image forgery detection techniques are based on PCA (Principal Component Analysis)[150] and the DCT (Discrete Cosine Transform) [45]. In summary, both techniques first apply the transform to each (usually sized 8×8) block in a sliding window fashion over the image. Then the resulting coefficients are sorted alphanumerically, where matching row numbers indicate blocks within copied regions.

ELA & BAG The reversal of the steps of JPEG compression also proves to be useful in ELA (Error Level Analysis) and BAG (Block Artifact Grid). For example, the BAG method and ELA both derive the compression type or level from within each of the individual blocks directly (again in a sliding window fashion.) In ELA, we subtract the recompressed image from the original one, highlighting the inconsistencies in the image which indicate fraud. (This will be described in more detail later.) In BAG, tampered regions are identified by having a different Q-table obtained from the histogram of each block coefficient DCT. To be more specific, the steps are summarized below:

- divide the image into 8×8 blocks (as used in JPEG compression)
- take the DCT of the blocks (using an 8x8 DCT matrix and matrix multiply as done in JPEG compression)
- make a histogram of the (quantized between -257 to 257) DCT values for each of the 64 locations of 8×8 blocks (the number of blocks is equal to the number that can fit into the image (eg. round(nrow/8)·round(ncol/8)/ (8×8)) so the number of values in each histogram is equal to the number of blocks
- take the FFT of the histogram of each of the 64 frequencies to get the periodicity and then power spectrum (absolute value) to get peaks
- calculate the number of local minimums of the filtered second derivative (extrema). This is the estimated Q value of the Q-table
- Once we get a Q estimate for at least 32 Q values in the table, we use the estimated Q to calculate the block artifact for each image block using the equation for the block artifact [181]

The segmentation mask is formed by plotting the compression level at each point in the matrix. In the proposed implementation, the mask from ELA is summed with the mask from BAG element-wise, enhancing the any detected variability.

2.5.2 RLRN

A summary of the RLRN algorithm's [110] basics steps are to first convert each image to greyscale, then de-correlate each image (subtract it from another copy of itself shifted by one pixel in each direction), and then count the total *run lengths* (or the number of the same pixel value in a row), for each color channel and each direction, for each image. These totals are stored in a vector with the element number representing the length. The ones that have the most (usually the top sixty) are used to form a feature vector for that image. These image vectors are then to be trained by a classifier as either *tampered* or *authentic*.

2.5.3 CNN, R-CNN, FASTER-RCNN

R-CNN forms the base of the model used for ML-based image fraud detection.

Specifically, Regional CNN (R-CNN) does the following: During training, a set of candidate regions of interest (RoIs) is generated, each possibly containing tampering, and then checked against ground truth, is as follows:

```
image data ->
generate RoIs ->
CNN layers ->
dense layers ->
tampering classification
```

Various refinements exist that make the generation and checking of the numerous RoIs more efficient. These include Fast R-CNN and Faster R-CNN.

2.6 Experimental Design

2.6.1 Classic Combined Methods

In the proposed combined PCA method, nine characteristic features that typically represent noise in image blocks within tampered regions are extracted. These are formed by a few basic operations such as taking sums and averages of certain sub-blocks as described in [90] [16]. Since these are smaller in number, it takes less time to identify copies in the sorted array. They are also more robust and therefore false positive predictions occur less often than using PCA alone.

The JPEG Quality-table is used in image compression to preserve low-frequency information and discard high-frequency (noise-like) detail [157]. The main difference between the PCA and DCT methods is that DCT also applies the inverse of JPEG compression and is better for detecting changes in JPEG images. The proposed implementation of DCT improves upon the original in [45] by dividing the blocks by either the JPEG chrominance or luminance matrices, rather than a matrix of random numbers. This helps bring out the artifacts since these are some of the most common image filters. Further, both the robust PCA and DCT methods have been parallelized.

2.6.2 Multi-stream Faster R-CNN

Subsequent work [186] improved the accuracy of image forgery detection over previous methods using a bi-linear approach. A combination of techniques which simultaneously examine both the RGB image content and noise information [102] is employed for tampering detection.

```
image data ->
RPN: generate candidate RoIs ->
conv. layers ->
stream1=RGB, stream2=Noise ->
dense layers ->
```

```
tampering classification
```

- stream 1, the RGB channels, are able to capture inconsistencies at tampered boundaries.
- stream 2, models local Noise, which may vary between the target image and the spliced portion

The output of the CNN is fed both into the stream1 and stream2 filters, both of which become features, and are combined in an outer product operation [93].

Our method instead uses ELA for the second stream.

```
image data ->
RPN: generate candidate RoIs ->
CNN layers RoIs ->
stream1=RGB, stream2=ELA ->
dense layers ->
tampering classification
```

The second stream is the JPEG Compression Stream, which is essential. Each 8×8 block in an image will have a Quality level helping us determine the localized region of interest. The pasted section will likely have a Quality level substantially different from the other parts of the picture, as described in the next paragraph 2.6.2.

JPEG COMPRESSION STREAM The Error Level Analysis (ELA) output is a grey-scale heat-map image that is created as follows: One saves the image at a slightly lower JPEG Quality level, reads it back in, and computes the pixel-by-pixel difference within 8×8 blocks from the original image. Since image regions with lower Quality in the original image will degrade at a higher rate when compressed, subtracting the decompressed image from the original image gives the difference in Quality levels in each block. Image blocks that originally had lower Quality levels will have the highest error and brightest color in the grey-scale output heat-map since the pixel values are higher. The steps of ELA are summarized below:

- read in image as JPEG
- write image as JPEG with Quality lower level (eg. 90)
- read in compressed image (decompress)
- take absolute value of the difference between the decompressed image in step 3 and the original image in step 1

The JPEG compression stream will have an input of the ELA or the combined BAG and ELA maps of the image to provide additional features of manipulation as shown in 3.

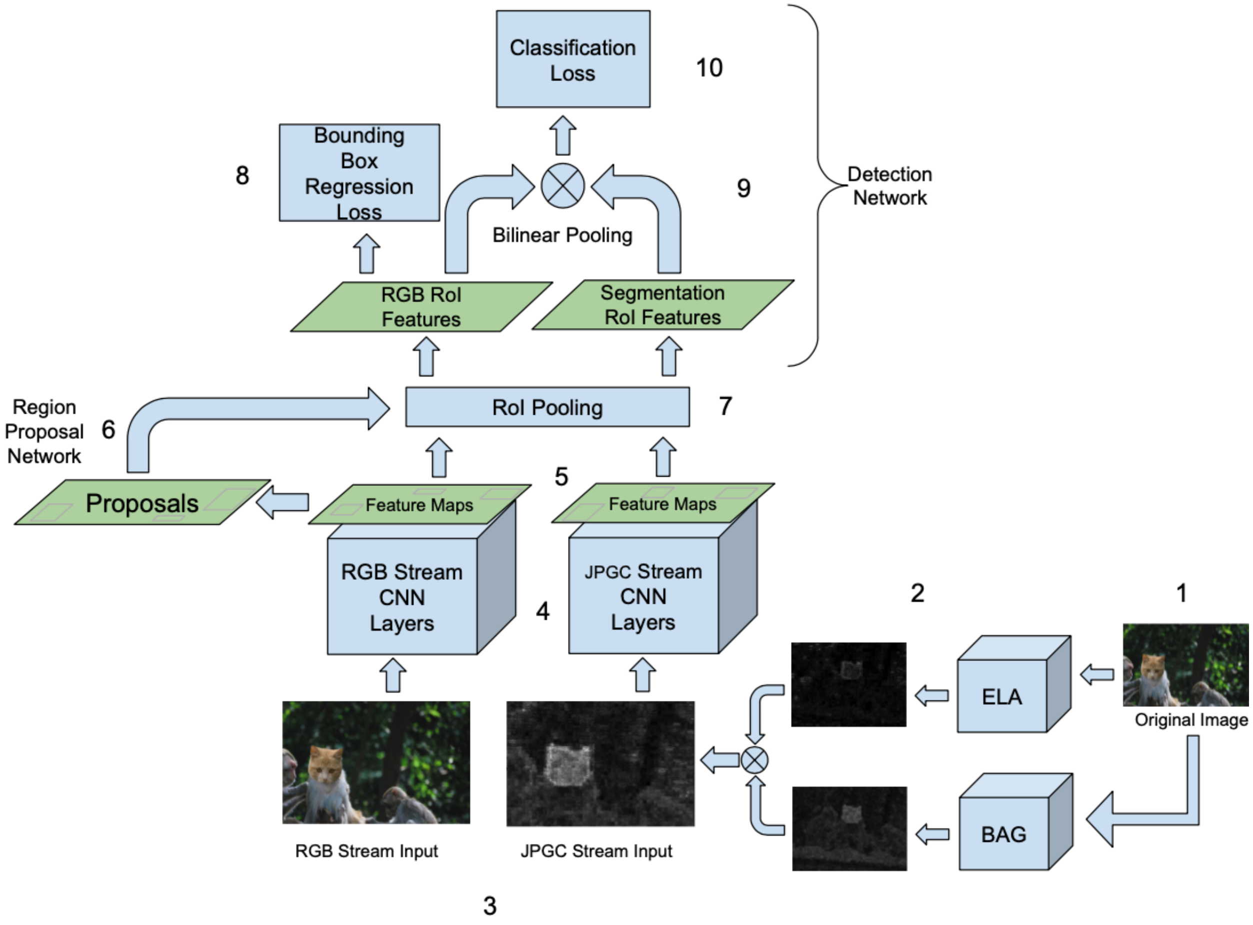


Figure 3: Multi-Stream Faster R-CNN Block diagram: 1. The original image is input into each filter 2. Both output error maps are combined 3. The original image and the output of ELA/BAG are each input into a separate stream of CNN 4. RGB stream and JPGC stream 5. The last layer of the CNN outputs anchors with multiple scales and aspect ratios which are used for the RPN to propose regions. 6. Only the last layer of the RGB stream CNN is used as input to the RPN, so both streams share these region proposals. 7. The ROI pooling layer selects spatial features from each stream and outputs a fixed length feature vector for each proposal. 8. The RoI features from the RGB stream alone are used for the final bounding box location prediction. 9. Bilinear Pooling is used to obtain spatial co-occurrence features from both streams. 10. The final predicted classes are output from the FCN and soft-max layers using sparse cross entropy loss.

RGB STREAM Since the RGB stream alone has been shown to be highly accurate in detection of manipulated regions [186], only this stream provides the region proposals of the RPN layer. Before being combined with the JPEG compression layer for RoI (Regions of Interest) pooling, the RoIs produced by the RPN of the RGB stream are used for bounding box regression to provide a localized box around the region including detection accuracy measures. After the two streams are fused through multi-stream pooling, they are input into the fully connected layer for manipulation classification [186].

RPN LOSS The loss for the RPN network is defined as

$$L_{RPN}(g_i, f_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(g_i, g_i^*) + \lambda \frac{1}{N_{reg}} \sum_i g_i^* L_{reg}(f_i, f_i^*)$$

- g_i represents the probability of anchor i being a manipulated region in a mini batch,
- g_i^* is the ground-truth label for anchor i to be manipulated
- f_i, f_i^* are the four dimensional bounding box coordinates for anchor i and the ground-truth, respectively
- L_{cls} denotes cross entropy loss for RPN network and L_{reg} denotes smooth L_1 loss for regression
- N_{cls} denotes the size of a mini-batch
- λ is a hyper-parameter to balance the two losses

MULTI-STREAM POOLING Multi-stream pooling is used to combine both streams while maintaining the spatial information between each of the pixels in the images being processed through each stream [186]. The output of the multi-stream pooling layer is $x = f_{RGB}^T f_{JPGC}$, where f_{RGB} is the RoI of the RGB stream and f_{JPGC} is the RoI of the JPEG compression analysis stream. The predicted classes of the Regions of Interest are output from the network's fully connected and soft-max layers and cross entropy loss is used for classification and smooth L_1 loss for bounding box regression [186]. The total loss function is the sum of all of the features as shown below:

$$L_{total} = L_{RPN} + L_{tamper}(f_{RGB}, f_{JPGC}) + L_{bbox}(f_{RGB})$$

where

- L_{total} denotes total loss
- L_{RPN} denotes the RPN loss
- L_{tamper} denotes the final cross entropy classification loss (based on the output of multi-stream pooling)

- L_{bbox} denotes the final bounding box regression loss
- f_{RGB} represents the RoI from the RGB stream
- f_{JPGC} represents the RoI from the JPEG compression stream

2.7 Evaluation Methods

2.7.1 Accuracy Calculation

MAP The model will be tested using the test function provided in Faster RCNN library [131] is based on the evaluation metric for PASCAL VOC project [42]. The AP score that this metric computes is based on the precision and recall. The precision measures how accurate the predictions are using the percentage of the correct predictions out of the total (where FP represents the number of false positive predictions and TP is the number of true positive predictions), as shown below:

$$Precision = \frac{TP}{FP+TP} \tag{2.1}$$

The recall measures how well all the positives are found in the test set (where FN is the number of false negative guesses), as shown below

$$Recall = \frac{TP}{FN+TP} \tag{2.2}$$

The Intersection over Union (IoU) is the overlap between the predicted bounding box coordinates and those from the ground truth image. For the PASCAL VOC project, the prediction is positive if the IoU is greater than 0.5. Average Precision (AP) is calculated based on an average of the 11-point recall values. Specifically, the AP is calculated as shown below,

$$AP = \frac{1}{11} \sum_{r \in 0.0,\ldots,1.0} AP_r \tag{2.3}$$

$$= \frac{1}{11} \sum_{r \in 0.0,\ldots,1.0} P_{interp}(r) \tag{2.4}$$

where $P_{interp}(r)$ is the precision at the recall value r.

F-SCORE The results are also based on the F-score as in 2.7.1, which is a harmonic mean of precision and recall (sensitivity):

$$F - score = \frac{2 \cdot (precision \cdot sensitivity)}{(precision + sensitivity)} \tag{2.5}$$

The precision measures how accurate the predictions are using the percentage of the correct predictions out of the total. It is calculated using the FP which represents the number of false positives, and TP which is the number of true positives, as in Equation 2.6:

$$Precision = \frac{TP}{FP+TP} \tag{2.6}$$

The recall measures how well positives are found, where FN is the number of false negatives (those ground truths which were not detected), as in Equation 2.7

$$Recall = \frac{TP}{FN+TP} \tag{2.7}$$

2.7.2 Datasets & Prepartion

- CASIA 1 (2013) [39]: 800 (authentic) + 921 (tampered) 374×256; Splicing with pre-processing, JPEG format images
  - Only images with clear bounding boxes were used for testing so that it would be easy to make a fair judgement of the overall prediction accuracy on this dataset.
  - TIFF images were converted to JPEG before being input into the ELA function
- CASIA 2 (2013) [39]: 7200 (authentic) + 5123 (tampered) 320×240 to 800×600; Includes splicing with pre-processing and/or post-processing, TIFF/JPEG/BMP image format
  - Only images with clear bounding boxes were used for testing so that it would be easy to make a fair judgement of the overall prediction accuracy on this dataset.
- CoMoFoD (2013) [158]: 5200 512×512 ; Includes color images, copy-move, JPEG/PNG format
  - The PNG images from CoMoFoD were converted to JPEG before being input into the ELA function and the machine learning algorithms.
- COVER (2016) [173]: 100 (authentic) + 100 (tampered) 400×486; Includes copy-move forged images with annotations
- The Image Manipulation Dataset - CMFD (2014) [27]: various 3440; Includes rotation, scaling, JPEG compression, noise
- image database constructed from the PASCAL VOC data (2012) [41]: 1,464 images for training, 1,449 images for validation and a private testing set; Pascal VOC is a collection of datasets for object detection (class recognition), and most commonly for benchmarking.

Table 16: Train/Test Data Setups

Dataset	Test/Train	Train Steps
Synthetic	5k/5k	45k
COVER	10/100	15k
CASIA	50/2886	25k
CoMoFoD	15/143	20k

Table 49 provides the Test/Train distributions and number of Training steps used respectively, for each dataset.

The test image sets from each database were randomly selected from the full set of images before training.

2.7.3 Software & Hardware

Please, see my the corresponding folder in the Image Fraud detection Github repository for the code to the specific algorithm [177]:

- **RLRNpar** folder: This is code uses a chroma-based method of feature analysis using Run Length Run Number (RLRN) endcoding. It trains a GLM model based on this feature array from a user input of an entire database of images. (For example, when trained with only 220 images it will then be able to detect whether an input image is has any type of image fraud or not with 97% accuracy.) (see README with subdirectory for more information)
  - Use this code for detecting just whether an image is fraudulent or not as it is the fastest, easiest to use and most accurate
- **BAGlocalization** folder: This code uses the Block Artifact Grid (BAG) to measure differences in the compression rate and quality factor of image blocks used in JPEG compression. Those blocks with a different factor are highlighted in red/yellow in the output heat map.
- **ELA** folder: This contains a code to run an ”Error Level Analysis” on JPEG images to determine the difference in compression levels throughout the image and localize forgery.
- **combinedLocalization** folder: This contains a code to combine BAG extraction and ELA with calls to rmBoxAvErrors.R and zeroOne.R to produce a highly localized output image. rmBoxAvErrors.R removes boxes with average errors less than a threshold to remove false positives, while zeroOne.R writes the leftover pixels to 1 or 0 using another threshold.
  - NOTE: Use this code to localize image forgery after determining the image was forged (and converting to JPEG) to help get the best localization approximation. First, be sure the image is determined by RLRNpar.R to be spliced by inserting a piece of another image (not by copy-paste from the same image).

- **pcaCProbust** folder: This contains a working combined intensity based and PCA algorithm (under the colder folder) for image fraud, as well as multiple test images and output (see README with subdirectory for more information)
    - NOTE: Use this code for localizing the actual copied region (of images with copy-paste forgery) as it is the fastest, easiest to use and most accurate
- **RLRNlocalization** folder: This contains an experimental working code for localizing tampered regions based on RLRN (see README with subdirectory for more information).
    - NOTE: Use this code for localizing tampered regions that are NOT copy-paste image fraud (tampered regions inserted from another image).
- **dctCPtested** folder: This contains a working DCT quantization algorithm (under the code folder) for copy-paste image fraud, as well as multiple test images and output (see README with sub-directory for more information).
- The code for the Multi-stream Faster RCNN was based on the Python version with VGG16 by Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun (Microsoft Research) [131]. The modified code is still on the cloud CPU which expired, unfortunately.
    - The model was implemented in Python on a Quadro 6000 GPU.

2.8 Improved Methods & Results

The author carried out each experiment below and recorded the results. The number of False Positives (FPs), where all True Positive boxes were localized in the test images for paralellization tests. The Test/Train distribution for each dataset, the number of train steps, and AP score were recorded for the Multi-stream Faster RCNN. For other ML algorithms the percentage correct (authentic or tampered class) is recorded.

2.8.1 Improved Classic Methods

Improved, Parallelized DCT & PCA It was found that running the sliding window with up to sixteen parallel cores, applying these classic, CPU-based (individually) algorithms is much more feasible since the runtime decreases significantly. If *n* CPU cores are used, this is done by beginning the operation from *n* equally distributed points in the image and combining the results at the manager node. The rows that overlap can easily be averaged.

Table 17 shows how the overall runtime of the PCA and DCT methods can be reduced by over five times, with little to no loss in accuracy. The first column shows the number of threads, while the second and third columns show the runtimes and number of false positives boxes by the modified DCT approach, respectively.

# of CPUs	time (s)	FPs
1	500	0
4	150	0
8	120	0
16	88	0

Table 17: DCT Parallel

Approximately the same results were obtained with the modified PCA algorithm since it also applies a transform in a sliding window fashion across all image blocks. Again, a shift of size one pixel is used. These tests were done on example images from each of CASIA [39], CoMoFoD [158], and COVER [173] which are a few of the most popularly used databases.

Several techniques can also be combined together to out-perform the stand-alone techniques. For example, the two middle images in Figure 4 show an example of how it will filter out most of the FPs by taking the nine *characteristic features* of the PC blocks from the normal PCA algorithm. Black or white covered boxes overlap detected tampered regions. (The first two images that are to their left are their originals containing pasted boxes on top and a pasted female walking in the bottom one.) The third image on the bottom and top shows the result of the application of the modified DCT method, where the box transform is divided by the luminance matrix.

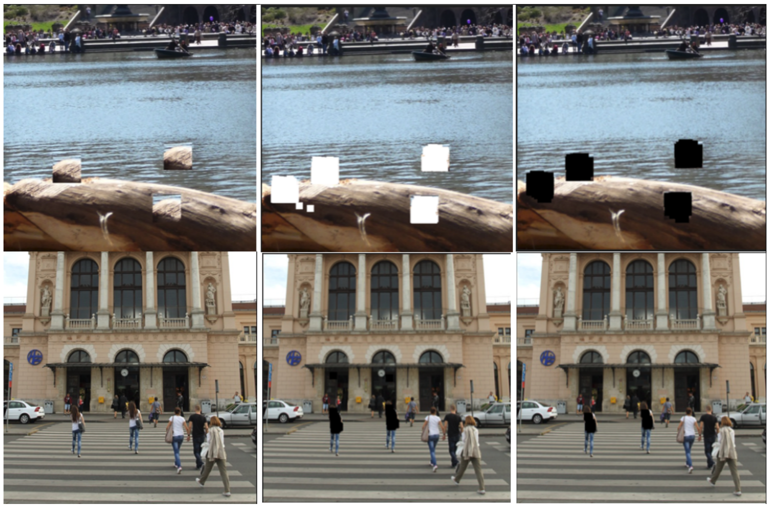


Figure 4: Column 1: Tampered input image (copy-paste fraud), Column 2: combined PCA algorithm output, Column 3: modified DCT algorithm output

Some methods work better than others depending on the image.

COMBINED BAG/ELA Finally, the segmentation masks produced by BAG and ELA will be summed element-wise to enhance the distinction between the tampered and authentic regions. This is done by applying a threshold to the total pixel value for each element. The final output (as shown in the bottom right in Figure 5) is higher contrast since the sum of the values produced by each method is a better estimate of the actual tampering level. These techniques are most useful for cases where resources for more powerful ML is unavailable or unnecessary.

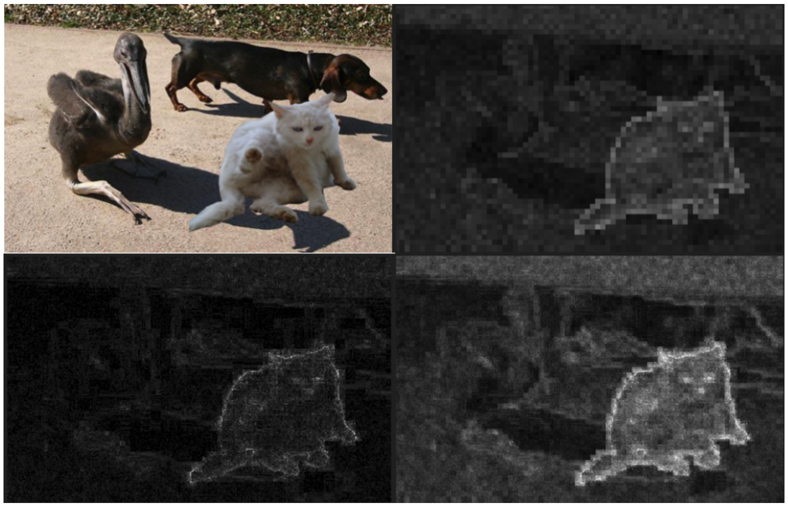


Figure 5: Top left: Tampered input image (copy-paste fraud), top right: BAG algorithm output, bottom left: ELA algorithm output, bottom right: ELA/BAG combined output (Image source: CASIA [39])

2.8.2 IMPROVED ML METHODS

PARALLEL RLRN ML methods are often far superior to the classic methods because they are typically trained on databases of thousands of images. On the other hand, extracting the run-length feature in algorithms like RLRN is time-consuming since it must process every pixel in an entire image for each feature vector. Since this can take seconds to compute for *each* image, it will add up when there is a whole database to compute. Parallelization is done by sending groups of images to each node on a cluster.

Here, the original implementation [110] was improved by allowing a (user input) number of parallel workers to distribute groups of images to compute the RLRN feature vectors. A super-linear speedup was obtained with a cluster of four lowering computational time from over an hour (in serial) to only a minute and a half on the CASIA [38] image database. This may be due to the fact that there are mutliple for-loops within RLRN feature vector calculation (including checking each row, column, and 1 to n pixel runs in each direction. Recall from section 1.2.4,

After applying the RLRN algorithm, GLM is used to achieve 97 % accuracy in prediction for each image as either tampered or authentic class for the modified implementation.

Also interestingly, even if we were to use an SVM for the classifier (as in the original implementation) with SA and a cluster of four nodes, a super-linear speedup can be achieved for this part of the algorithm as well. This improves

Table 18: Performance Comparison to Other Models

Dataset	MS-FRCNN	RGB-N	RGB Net
COVER	0.82	0.437	0.39
CASIA	0.69	0.408	0.39
CoMoFoD	0.59	-	-

upon the original implementation [110] using only a serial SVM which takes over two hours to compute with lower accuracy (since it does not se GLM). However, in this the modified version of the RLRN, SVM will be replaced with GLM as it takes a small fraction of the time and predicts with higher accuracy.

Multi-stream Faster RCNN

Synthetic Dataset Tests The network was first trained and tested on a self-made spliced image database constructed from the PASCAL VOC data [41]. It was created by digitally selecting the random objects by their pixel maps provided in each dataset, pasting them into another image, and moving the new object annotation with it.

The accuracy was over 90% on the synthetic dataset. Top methods such as, Yan et. al. [176] also created a synthetic dataset using PASCAL VOC 2012 and obtained 87%. They used a CNN-based deep architecture, which consists of three feature extraction blocks and a feature fusion module.

Official Image Manipulation Dataset Tests Second, it was fine-tuned and tested on a few classic image manipulation datasets, which have been highly re-used in literature. These include subsets of CASIA 1 & 2 [39], CoMoFoD [158], and COVER [173]. Only those images from CASIA 1 & 2 with clear bounding boxes were used for testing so that it would be easy to make a fair judgement of the overall prediction accuracy on this dataset.

The test sets were randomly selected from the full set of images before training. Since training beyond 45k steps did not improve accuracy, each training session was stopped prior to this point. (Lower numbers of steps were required for those datasets with less images.) The PNG images from CoMoFoD and the TIFF images from CASIA were converted to JPEG before being input into the ELA function.

Table 18 shows the performance comparison to the single stream version (RGB Net) and the bilinear version with the second stream as the noise features (RGB-N) [186].

Many DL networks built for image fraud only tested the copy-paste forged images. For example, BusterNet [174] was a past top performing model on the copy-move-*only* subsets of the datasets. Addtionally, Kumar and Bhavasar obtained slightly higher scores on CASIA and CoMoFoD dataset to 80.3 % and 78.8%, but only tested the copy-paste forged images, as well. However, similar to the differences in evaluation metrics, the wide variety of test set altercations makes it difficult to compare exactly to very many other models.

Figure 6 shows additional examples of visualizations of results from tests on images from the CASIA, CoMoFoD, and Image Manipulation Datasets [27], respectively. Clearly, each prediction with the ELA stream (in the top row) is better than the single model alone, because the second stream helps to remove false positives by providing additional training features based on the level of compression of the manipulated regions.

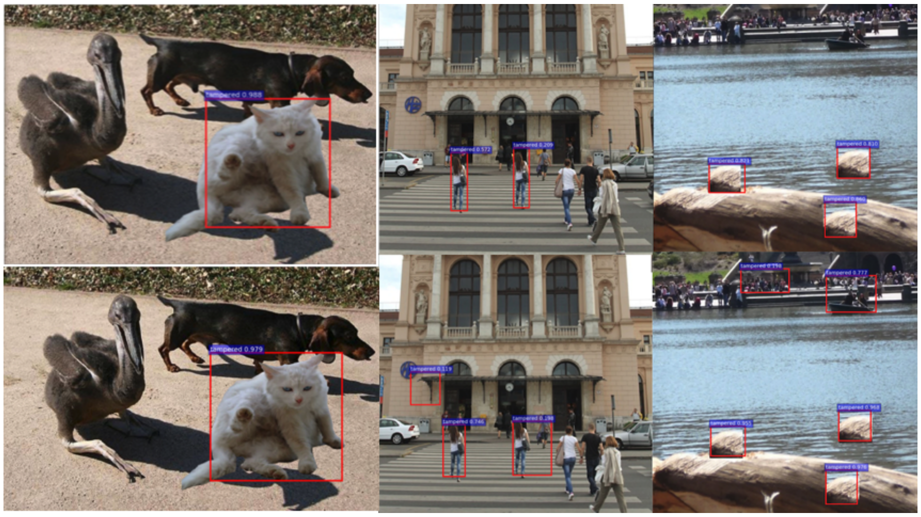


Figure 6: Top: Output of the Multi-stream Faster-RCNN Model with ELA on PNG Images, Bottom: Output of the Faster-RCNN Model

2.9 Conclusions

2.9.1 Multi-stream Faster R-CNN Discussion

Both the Multi-stream Faster R-CNN version were able to effectively localize tampered regions from the traditional image databases [39], which shows that this proposed model is useful for detecting image fraud. The predicted accuracy of the tampered region, the proposed coordinates of the bounding box, and the FP rate are more accurate in the Multi-stream Faster R-CNN version with ELA.

Further, although the test set is likely slightly different, the score is higher than those in [186]. Based on a comparison of the output images, it is also likely to be robust to BAG, ELA, noise analysis, DCT, and RLRN detection [108] on any image while having much clearer and more precise results. This implementation also has the ability to be generalized since non-JPEG images can be converted to JPEG prior to being input to the model for ELA without loss of prediction accuracy.

The previous methods also lack a feature to place a bounding box with its accuracy over the localized region. Older methods normally use a sliding window of feature maps derived from fixed size box of image pixels to test for manipulated regions. This network uses bounding box regression on various anchor box sizes to estimate the probability of a region being tampered allowing us to simultaneously capture more information within regions of the image.

2.9.2 Future Work

Improving Multi-Stream Faster-RCNN Further work with Faster-RCNN or any modification is not recommended based on the results that have been obtained with YOLOv4, YOLOv5, and YOLOR in highly similar image applications in the third section here 3. However, section 3.10.1 also discusses some of the major improvements that would need to be made to the Multi-stream or Faster-RCNN model for it to be anywhere near comparable to any of the YOLO versions frameworks (see paragraph 2.9.2 below) for basically any type of object detection.

YOLO Proposal based, two-stage detector methods like Faster-RCNN require higher computational costs in time and resources, while on the other hand YOLO [125] is a proposal free approach that can be easily adapted into real-life settings for quick analysis [68].

YOLO (also known as *You Only Look Once*) [125] is a one-stage, real-time object detector. It is an example of another alternative object detection network that could solve the problems with classic methods by extracting various complex features within the localized region. (See section 3.5.1 for a more complete discussion of YOLOv3, YOLOv4, YOLOv5, and YOLOR.) This is especially the case for the latest version - YOLOR [168] (also known as *You Only Learn One Representation*) which is currently both the fastest and most accurate state-of-the-art object detection network. YOLOR is discussed in more detail in the next Part in paragraph 9.

It also far outperforms both Faster-RCNN and the Multi-stream Faster RCNN in the similar image processing task of mitosis counting, which will be discussed in the next part and is shown in the Results section 3.8. Better yet, it is likely to further out-perform the former object detection models when used for image fraud detection because it is and ideal model for the data-sets common in image forgery. For example, it uses many different data augmentation methods which can help solve the issue of the smaller data-set sizes typically available in image forgery by creating more training examples for the network to learn from Subsubsection 2.7.2.

Since the models adapted to mitosis counting were originally useful in image fraud detection, it should be the other way around as well. Both methods require localizing details and textures which can be difficult to identify by the human eye alone. Both have historically used similar methods/algorithms of feature extraction, as described in Subsection 3.3. Use of YOLOR or YOLOv5 shows the strongest potential in the application to image fraud detection, as discussed in Subsubsection 3.9.6.

A few groups have already tested one of the older YOLO version, still with impressive results. For example, copy-move attacks are common in object-based video forgery, and most common methods are to slow, so a real-time detector is in great need. Raskar and Shah used YOLOv2 applied to video forgery detection to localize complex copy-move attacks such as rotation, scaling, and flipping. They obtained a 0.99 confidence score on benchmark video forgery datasets (eg. Surrey University Library for Forensic Analysis (SULFA) created [118]), outperforming state-of-the-art methods [124].

YOLOv3 has also been adapted for the real-time detection of diseases in tomato plants by Liu and Wang [95]. This implementation reduces the huge loss caused by diseases and pests in crops.

TRAINING & TESTING ON OTHER DATASETS The authors of the highest scoring implementations for the Faster-RCNN model with the Noise stream obtained their score by combining the training sets for multiple benchmark image forgery contests, fine-tuning with a new database after a set number of epochs [187]. Therefore, it will be important to try this and compare the score.

Additionally, there are other bench-mark contest datasets available that should be prepared to test alone, as well as in combination with the other. Some of these are listed below:

- Columbia color (2006) [111]: 183 (authentic) + 180 (tampered); Includes image splicing, TIFF format color images
- Columbia gray (2004) [60]: 933 (authentic) + 912 (tampered); Includes splicing, BMP format greyscale images
- INRIA Copydays (2008): 1642; Includes cropped images, scaling, and JPEG compression, combined attacks
- Dresden [51]: 25,137; Includes images taken from different camera models, each with different quality levels and file formats

It would be especially useful to pre-train the YOLOR model with the other datasets combined and then fine-tune with only the training set portion of the full contest dataset that we are testing on.

3 DEEP LEARNING FOR MITOSIS COUNTING

3.1 ABSTRACT

It is estimated that breast cancer incidences will increase by more than 50% by 2030 from 2011 [9]. Mitotic count is one of the most commonly used methods of assessing the progression for every patient diagnosed with invasive breast cancer [9]. Although mitotic count is the strongest prognostic value [86], it is a tedious and subjective task with poor reproducibility, especially for non-experts [162]. The following study will show how the Multi-stream Faster RCNN can be adapted for mitosis counting in breast cancer detection with the second stream having an input of the UNet generated compression level mask. Then it will be shown that the speed and accuracy of newer state-of-the-art methods of object detection such as YOLOv3, YOLOv4-scaled, and YOLOv5 [14], as well as YOLOR, have been proven to even surpass the Multi-stream Faster-RCNN in speed and accuracy in various domains of object detection. Multiple methods of increasing the size of the data for improvement in learning, such as augmentation, as well as adding more scanners and image datasets are some of the other minor techniques used to improve the score overall. Results show that the highest F-score on the mitosis counting dataset MITOS-ATYPIA 2014 challenge [136] that can be achieved is up to 0.95 and at least 0.96 on the MITOS-ATYPIA 2012 challenge [137] with YOLOR and using a combination of methods of data augmentation to increase the size of the training set.

3.2 Introduction & Significance

3.2.1 Mitotic Count & Issues

The Nottingham Grading System (NGS) is recommended by various professional bodies internationally (World Health Organization [WHO], American Joint Committee on Cancer [AJCC], European Union [EU], and the Royal College of Pathologists (UK RCPath) [121]. It says that tubule formation, nuclear pleomorphism, and mitotic index should each be rated from 1 to 3, with the final score ranging between 3 and 9. This is divided into three grades: Grade 1, score 3–5, well differentiated; Grade 2, score 6–7, moderately differentiated; and Grade 3, score 8–9, poorly differentiated [8].

When pathologists need to make this assessment of the tumor for mitotic count, they start by finding the region with the highest proliferative activity. The mitotic count is used to predict the aggressiveness of a tumor and is defined in a region from ten consecutive high-power fields (HPF) within a space of $2mm^2$. Mitosis also often have a low density and can look different depending on whether it is in one of the four main phases: prophase, metaphase, anaphase, and telophase, as shown in Figure 7. The shape of the cell itself differs significantly for each phase. For example, when in telophase it is split into separate regions even though they are still one connected mitotic cell.

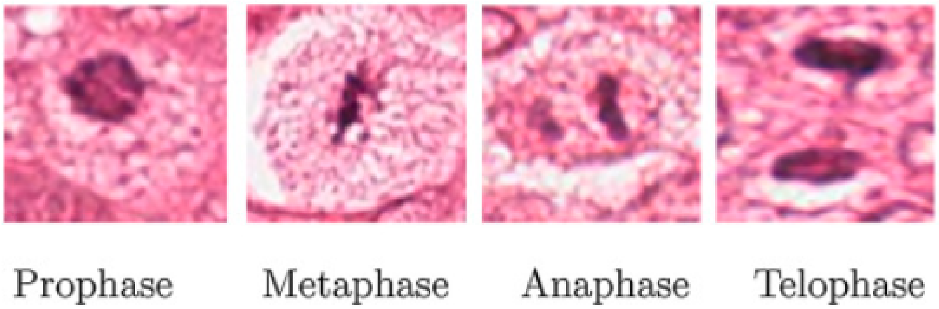


Figure 7: The Phases of Mitosis [141]

Apoptotic cells (or cells going through preprogrammed cell death) and other scattered pieces on the slides can also easily be confused with mitoses, having a similar dark spotty appearance. Further, mitotic nuclei often resemble many other hyperchromatic cellular bodies such as necrotic and non-dividing dense nuclei, making detection of mitosis more difficult on tissue [161].

The variation in the process of obtaining the slides using different scanners and different preparation techniques may also make distinguishing cells more exhausting. Worse yet, pathologists can get tired and it can make it harder to make proper judgement on slides when trained pathologists need to examine hundreds of high power fields (HPF) of histology images, in a short amount of time.

The Figure 8 shows the examples of mitotic figures versus similar non-mitotic figures. Mitotic figures are shown in the top row, while non-mitotic figures are shown in the bottom row.

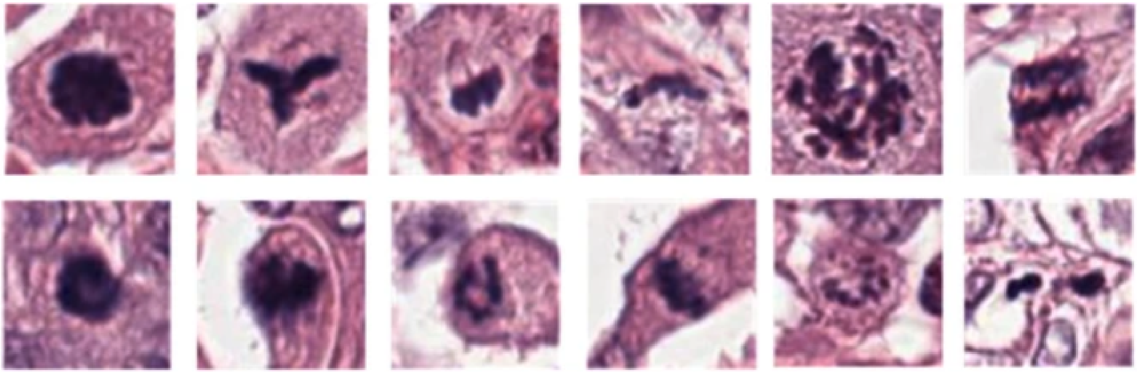


Figure 8: Mitotic (Top Row) versus Non-Mitotic (Bottom Row) Figures 7

The increasing breast cancer incidences calls for a more time and cost efficient method of prognosis, which could later even help to provide care to impoverished regions. Automatic image analysis has recently proven to be a possible solution [162].

3.3 Related Work

3.3.1 Automatic and Machine Learning Methods

The use and development of automatic detection methods of mitosis counting have gradually been increasing since the end of the 20th century in order to make doctors' jobs easier and more efficient [114]. Due to the recent progress, a large amount has became available for use in the medical studies. Machine learning has helped to discover new characteristics of cancer by sorting through more image data than humanly possible and simultaneously analyzing millions of image pixels. For example, in the field of histopathology, machine learning algorithms have been used for analysis of scanned slides to assist in tasks including diagnosis [77]. The use of computing in image analysis may reduce variability in interpretation, improve classification accuracy, and provide clinicians (or those in training) with a second opinion [77]. Existing methods use either handcrafted features captured by specific morphological, statistical, or textural attributes determined by pathologists or features are automatically learned through the use of convolutional neural networks (CNN).

3.3.2 Manual Feature Extraction

Similar to earlier image fraud detection techniques, discussed in **??** automatic detection methods of mitosis mainly consisted of machine recognition based off of manually designed features. Manually designed features are required to be described by a human to represent the characteristics of mitosis. This is usually based on a human's professional domain knowledge of appropriate mitosis textural or morphological features. On the other hand, with newer deep learning methods these features are able to be discovered by the machine, such as with the use of neural networks.

Some former methods also have combined machine learning with manually-designed features [142]. These usually use two main steps: segmentation and classification [114]. In the first step candidate regions are segmented to decrease the size of the input image locations that will need to be tested (usually by using maximum likelihood estimation, threshold segmentation, and watershed segmentation). In the second step the classification algorithm is trained specifically to detect the hand described features to extract mitosis from non-mitosis in the pre-segmented region. For example, [154] first used MLE (Maxiumum Likelihood Estimation) to extract pixel-wise features based on color information captured by MLE. Then they distinguished mitotic regions from non-mitotic regions using the extracted textural features obtained from CLBP (Completed Local Binary Pattern). And, lastly they used an SVM (Support Vector Machine) classifier to classify extracted feature vectors. [116] used the maximization of relative-entropy between the cells and the background to segment and then used a RF classified to discriminate mitotic from non-mitotic regions. Zhang et al. proposed a method that fuses handcrafted features based on four indices of the GLCM (Gray Level Co-occurrence Matrix) and deep features based on natural image knowledge transfer to segment mitotic cells [183].

Irshad first analyzed statistics and morphological features in specific channels of various color spaces which are used to assist pathologists in mitosis extraction. Next, they then performed a LoG (Laplacian of Gaussian), thresholding, morphology, and active contour model on blue-ratio image to find and segment mitotic candidates. Candidates are selected based on first and second order morphological and textural features to be classified using a DT (Decision Tree) classifier [64].

3.3.3 Deep Learning Methods

With the help of their strong self-learning qualities, deep learning networks, especially neural networks have also been heavily investigated in medical image processing [145]. CNN's (Convolutional Neural Networks) have made a significant impact in machine learning for image classification, segmentation, object detection, and computer vision tasks [35]. Medical applications in particular, such as mitosis detection, cell nucleus segmentation and tissue classification tasks have also been popular tasks for CNN's. This is because pathological images are texture-like in nature, making them ideal task for a CNN to learn with their shift invariance and pooling operations. Deep learning methods often outperform traditional methods such as use the of handcrafted features alone since feature extractors and classifiers can be simultaneously optimized.

Although handcrafted features are domain-inspired for the particular application, the data-driven CNN models have the ability to learn additional feature bases that cannot be represented through any of the handcrafted features. Some approaches combine a CNN or deep-model and handcrafted features, in order to exploit the advantages of hand generated features and combine them with CNN. For example, Wang et al. used a cascade approach to combine both types of features. In order to minimize computational resources, they perform classification via CNN and handcrafted features separately. A CNN is applied on pixel-patches, and then the handcrafted features are extracted from the clusters of segmented nuclei. This yielded an F-measure of 0.7345 [171]. Saha et al. combined a deep learning architecture consisting of five convolution layers, four max-pooling layers, four rectified linear units (ReLU), and two fully connected layers with handcrafted features. ReLU activation is applied after each convolutional layer as an activation function and dropout is applied after the first fully connected layer. This implementation was used to obtain an F-score of 0.90 on the contest datasets [142].

CNNs are well-suited to learn high level features such as mitotic figures, which is likely what made these methods winners of the MITOS-ATYPIA 2012 challenge [137], the MITOS-ATYPIA 2014 challenge [136] 3.7.1, and the AMIDA 2013 challenge [163] [123]. These are the well-known mitosis detection challenge competitions which were held at conferences and provide a great resource to exchange information. They are publicly available datasets commonly used for mitotic detection of breast cancer research and will be further discussed in 3.7.1. Cireşan et al. won ICPR 2012 using deep max-pooling convolutional neural networks to classify each image pixel using a patch centered on the pixel as context. The simple CNN consisted of five convolutional layers with max pooling layer, and two fully connected layers [29]. A similar model has also been successfully used in detecting mitoses from the AMIDA13 challenge [88]. A multi-column neural network is used to classify image patches and generate the precise image descriptors.

One drawback is that although they have high classification accuracy, they can also have a high computational load. Further, pre-processing the full input image prior to applying the CNN can help reduce the work of differentiating mitotic background regions versus non-mitotic background regions. Therefore, a number of groups have tried to alleviate this issue by designing candidate extractors for regions prior to input into the CNN. For example, Li et al. used a feature-based region extractor plus two CNN stages to achieve an F-score of 0.78 on the ICPR 2012 dataset and an F-score of 0.66 on the ICPR 2014 dataset [88]. As another example, Chen et al. used a retrieval model prior to a three-layer FCN (Fully Connected Network) followed by a CaffeNet [69] to classify patches and achieved an F-score of 0.79 on the ICPR 2012 dataset [21].

3.3.4 Object Detection for Histopathological Image Analysis

On the other hand, it has now become well-known that a basic CNN alone lacks cell-level supervision and often requires limiting the size of the input image. This is done so that sub-image features can be learned from localized regions of the image rather than the full image context consisting of multiple objects as well as non-objects (regions of interest). Further, object detection or precise localization is actually a more common task than full-image classification in medical application. Consequently, new research has found a solution: the application of deep learning methods designed originally for object detection on ImageNet such as R-CNN, Faster R-CNN, and Mask R-CNN. These networks can actually be adapted to medical tasks in order to target other specific features or textures from within the image which have been deduced from a ROI (Region of Interest).

Lu et al. designed a cascade detection algorithm based on segmentation and classification which reached 0.83 on the ICPR 2012 data set and 0.58 on the ICPR 2014 data set. This cascaded CNN was based on UNet and consisted of three parts. First, UNet is used for segmentation to locate the candidate set of mitotic targets. Second, the cell nucleus is located by means of semantic segmentation to obtain accurate image blocks of mitotic and non-mitotic cells via a Vnet. Third, the cell image output block is used to train a CNN to do binary classification and this area is checked for mitosis [98]. Sebai et al. developed a multi-task deep learning framework for both object detection and instance segmentation tasks using Mask RCNN. First it is used for segmentation to estimate the mitosis mask labels for the weakly annotated mitosis dataset. This produces the mitosis mask and bounding box labels for training another mitosis detection and instance segmentation model for mitosis detection on the other dataset [144]. They obtained an F-score of 0.86 on the 2012 ICPR dataset and an F-score of 0.48 on the 2014 ICPR dataset. Rao used Faster-RCNN to achieve 0.96, the highest F-score in reported in the mitosis detection contests [123] by training on all three challenge datasets above. This is 6% more accurate than the previous high score of 0.90 achieved by the model proposed by [143].

3.3.5 Multi-stream Model For Mitosis Counting

Historically, the classification and detection models used for another field, *image tampering detection* 2, are similar to that of mitosis detection in HPFs. For both, their purpose is to localize regions of interest with different textures (nearly indistinguishable to the human eye). This is why histopathological image segmentation is also commonly used to extract and highlight objects of interest from the background of the image (with different textures) for further identification. For example, some of the segmentation methods used for mitosis counting or cancer cell identification have included thresholding (using Fourier descriptors, wavelets or Otsu), edge detection (using Canny and Sobel filters), or clustering (such as k-means, mean-shift, and K-nn) [79][156]. Therefore, it makes sense to test the Multi-stream version with mitosis detection, because it achieved the highest accuracy in image fraud detection [186]. Segmentation of medical images can be done with many different existing algorithms, such as UNet [135], which can be applied to the second stream of the model.

3.3.6 Advancing Object Detection Methods in Mitosis Counting

Additionally, there are multiple newer object detection networks such as YOLOv4-scaled, YOLOv5, and YOLOR [14] now far surpass Faster RCNN in both speed and accuracy when tested in many applications of object detection [68].

The addition of data augmentation techniques may also significantly improve accuracy over that of Faster RCNN, since it is included in the architecture of the later versions of YOLO model. For example, the Cutmix [182] (implemented initially in the YOLOv3 architecture) in EfficientDetdx helped improve results as well as achieve top scores on the Global Wheathead data [178]. Cutmix (which can work better than the simple MixUp and Cutout augmentation types) combines images by cutting parts from one image, and pasting them onto augmented image forcing the model to learn to make predictions based on robust features.

A number of research groups have also modified one of these two networks for other types of biomedical object detection in images with superior results. For example, Jiang et al. helped improve the accuracy of blood cell count using YOLO for a spatial attention feature extraction network [70]. Deep learning for object detcetion was used to detect diabetic foot ulcers [52] and for detection in endoscopic images [113]. YOLO has been used for both kidney localization and detection and localization of lung nodules in CT scans [82] [138].

These promising results from YOLO warrant further study with other new object detection applications in the medical field. Mitosis counting, in particular, is one specific area where this framework is highly applicable and where these advancements have not yet been tested. This calls for further investigation in this task.

3.4 Specific Aims

The first requirement of this study was to verify that UNet is a good framework to use to segment biomedical images for the second stream of the Multi-stream model. The purpose of this was to generate the segmented images which are not provided with the training data, as was the case with the Image Fraud detection task previously used with this model.

Once the segmented images are obtained, the first goal was to test whether a Multi-stream Faster RCNN with the segmentation stream is an effective method of mitosis localization on two of the most widely used mitosis detection contest datasets (ICPR 2012 and ICPR 2014).

The second goal was to test newer or more advanced object detection networks such as YOLOv3, YOLOv4-scaled, YOLOv5, and YOLOR, in comparison to the Multi-stream Faster-RCNN model in the mitosis counting task.

The third goal was to try a number of different methods of increasing the size of the training data to further improve prediction accuracy. This additional data preprocessing included adding images from multiple scanners, combining the two different contest datasets, multiple forms of augmenting images, as well as decreasing the shift in the sliding window for the original data generation.

3.5 Definitions & Background

Multi-stream Faster-RCNN The Multi-stream Faster-RCNN was described in Section 2.6.2.

UNet Segmentation Preparation of the data for training the second stream of the Multi-stream Faster R-CNN was be done by applying segmentation to the images for application to the segmentation stream with a UNet.

This is a 23-layer network is based on the work of Ronneberger et al. in 2015 [135], winner of ISBI Challenges in biomedical image segmentation and cell tracking [7]. The down-sampling side of the 'U' doubles the number of features in each step while the up-sampling side halves features in each step. The contracting path uses repeated pairs of 3×3 un-padded convolutions to capture image context before a rectified linear unit (ReLU), followed by a 2×2 max pooling operation with stride two. The symmetric expanding path helps to achieve localization and uses a 2×2 convolution and two 3×3 convolutions each followed by a ReLU.

3.5.1 YOLO

YOLO is a one-stage detector (eg. why its called 'You Only Look Once') so the localization mechanism is done in combination with classification instead of as a separate task as in Faster RCNN. The first YOLO version was originally written by Joseph Redmon in low-level languages, using a *Darknet* backbone. Then, YOLOv2 made improvements on top of YOLO including BatchNorm, higher resolution, and anchor boxes. YOLOv3 then built upon those two versions to include objectness score and levels of granularity to bounding box prediction, as well as connections to the backbone network layers. Unlike Faster-RCNN, YOLO uses a proposal-free approach which reduces the computational cost and allows for real-time detection. Most modern accurate models require many GPUs for training with large mini-batch size, while this network is optimized one GPU.

YOLOv4 and YOLOv5 Aleksey Bochkovskiy helped create YOLOv4, which surpassed YOLOv3, YOLOv2, and YOLOv1 in detection speed and accuracy. The main contribution in YOLOv4 architecture is that the modules do data augmentation internally. These are included in what they call *BoF (Bag of Freebies)* [14], which refers to blocks that improve performance accuracy of the detector without adding to the inference cost in prediction.

Although, a paper on latest fifth-generation YOLO update has not been published yet, it was built by Glen Jocher from the previous YOLO models. Also, from the fourth-generation model it includes a number of innovations including Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation, Mosaic data augmentation, CutOut, Random Erasing, DropOut, DropConnect, DropBlock regularization, and CIoU loss. The v5 model achieves state-of-the-art results at real time speed on the MS COCO dataset with 43.5% AP running at 65 FPS on ImageNet [35].

SCALED-YOLOV4 Aleksey Bochkovskiy helped Hong-Yuan Mark Liao to create YOLOv4-scaled surpass previous benchmarks in object detection such as EfficientDet (developed by Google Brain Research team) [153], as shown in the Figure 9. In Scaled-YOLOv4, the authors start out by training on a less augmented dataset first, then they add additional augmentations for fine-tuning once it has been trained for a while. They also use what they call ”Test Time Augmentations” where a number of augmentations are made on the test set so that they can be averaged across those for an additional boost non-real time results. The depth and number of stages are also scaled up for large images.

They designed YOLOv4-large specifically for cloud GPU, to achieve the highest accuracy for object detection. They then designed a fully CSP-ized model called YOLOv4-P5 and scaled it up to YOLOv4-P6 and YOLOv4-P7 for multiple GPUs [165]. CSP-ized means that it applies the concepts of Cross-Stage Partial Networks [167] used to reduce the computation required for CNNs.

YOLOR YOLOR stands for ”You Only Learn One Representation” was released in mid-2021 for object detection. It has a different author, architecture, and model infrastructure then the previous YOLO versions described above [169]. The main components that make this architecture functional are the CNN, the kernel space alignment, and prediction refinement. It was designed based on the concept that the brain combines implicit knowledge (gained from deep layers) from the past with explicit (clear straight-forward given) knowledge gained from new experiences. Then a general representation is used to refine the prediction. The CNN of YOLOR learns to predict multiple possibilities of the output instead of just one. YOLOR was proven to produce 88% faster results with similar accuracy to YOLOv4-Scaled, making it the fastest existing detection algorithm, as shown in Figure 9.

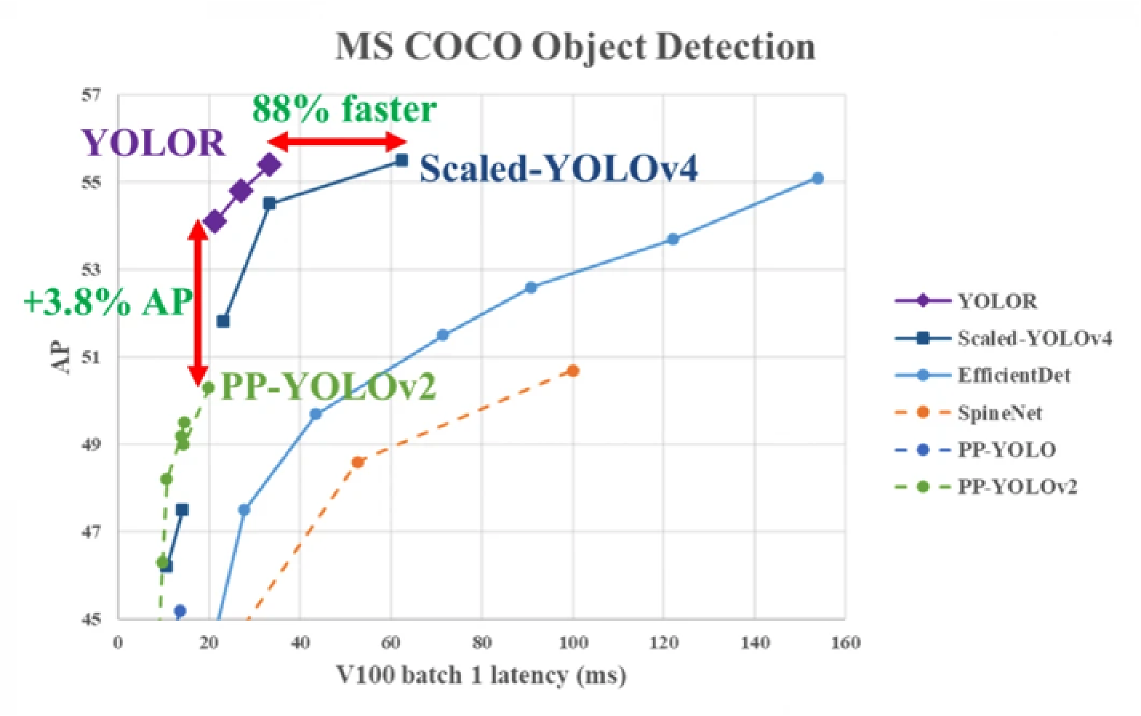


Figure 9: Performance of YOLO Models vs EfficientDet [169]

3.5.2 Data Augmentation

Image augmentation creates new training examples out of existing training data images in order to generalize to other situations. Using this technique allows the model to learn from a larger variety of changes in image brightness, scaling, size, rotation, and noise.

3.6 Experimental Design

3.6.1 Multi-stream Faster RCNN

The Multi-stream Faster-RCNN was described in Section 2.6.2.

UNet Segmentation Part of the process of preparing the data for training Multi-stream Faster R-CNN includes applying segmentation to the images for application to the segmentation stream. This was be done with UNet, as shown in 1 and 2 of the block diagram of Figure 10 which lays out the training process.

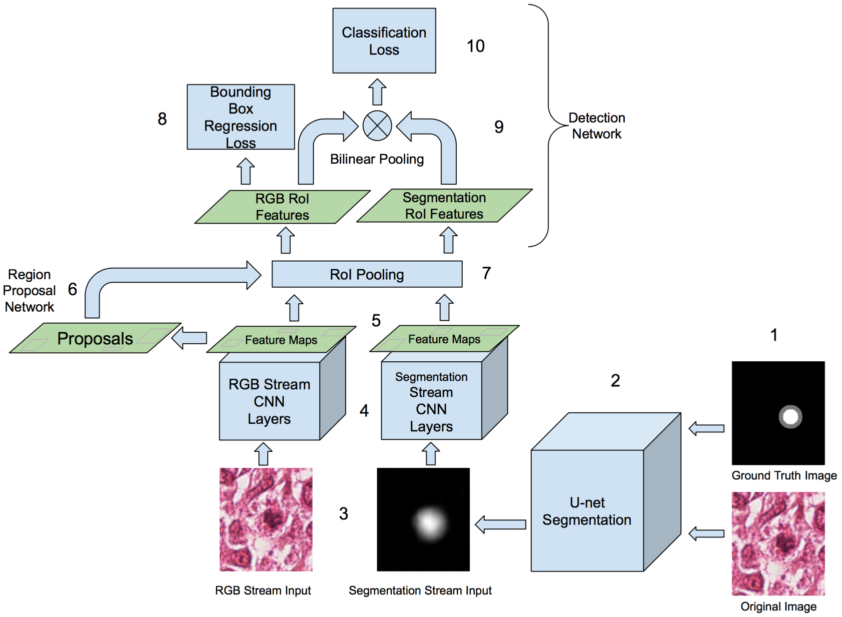


Figure 10: Multi-Stream Faster R-CNN Training Setup. 1. UNet is pre-trained using half of the RGB images along with the ground truth masks. 2. Segmented images are then produced by testing the complete set of Multi-stream Faster R-CNN training images on UNet. 3-4. Cropped RGB images are input to the RGB stream CNN, while segmented images are input to the segmentation stream input. 5. The last layer of the CNN outputs anchors with multiple scales and aspect ratios which are used for the RPN to propose regions. 6. Only the last layer of the RGB stream CNN is used as input to the RPN, so both streams share these region proposals. 7. The ROI pooling layer selects spatial features from each stream and outputs a fixed length feature vector for each proposal. 8. The RoI features from the RGB stream alone are used for the final mitosis bounding box location prediction. 9. Bilinear Pooling is used to obtain spatial co-occurrence features from both streams. 10. The final predicted classes are output from the FCN and soft-max layers using sparse cross entropy loss. [188].

Overall Multi-stream Faster-RCNN Set-up

3.6.2 Finding the Optimal Model & Configuration

Once the best augmentation combination was found, different numbers of epochs and versions were tested for each YOLO version model to find the optimal setup for speed and accuracy for this specific application. It was then used for the further testing with the training data such as the combined datasets. This also helps the performance capability, preferred setup, and overall usefulness in a real-world applications. These results were compared to Multi-stream Faster RCNN by using the same training and testing sets. Possible reasons for the favorable configuration will then be addressed in the discussion in detail section.

3.6.3 Combining YOLOv5m-p5 with YOLOR

Both YOLOv5m-p5 and YOLOR consistently produced the highest F-scores, but with different predictions. Also, YOLOR predicted its highest scores at quicker runtimes. Therefore, when the predictions made by YOLOv5m-p5 and YOLOR were averaged, both the results and runtimes could be optimized. This helps to refine the results without loss in efficiency because each of the models can be trained and tested in parallel on a separate cloud GPU.

3.6.4 Increasing the Size of Training Data

Multiple Scanners To test for the potential change in accuracy by adding data of multiple scanners, training was done with the model on the images from each scanner alone and testing on images from the same scanner on which it was trained. It was then be compared to the results of training on the combined set to see if adding data from the other scanner helps. Next, testing was be done by alternating the training and testing data to test on data from another scanner besides the one of which it was trained on. These tests may also be interesting or useful for realistic situations in which similar training data (eg. with the same scanner) for the image was missing. Training and testing on sets of combinations of each may also produce useful results to provide insight on which scanner achieves higher accuracy.

Multiple Databases Then, to assess the effect of combining data from multiple databases to the training set, the ICPR 2012 training set was mixed with the ICPR 2014 training set. If the predictions on the test images from one database alone are better when the model is trained with data from both then this will help us determine how useful this could be in real-life cancer detection. For example, we could continue to add data to the training set and keep updating the weights to get better results on any test set (or any real life patient who may have been using another scanner/database).

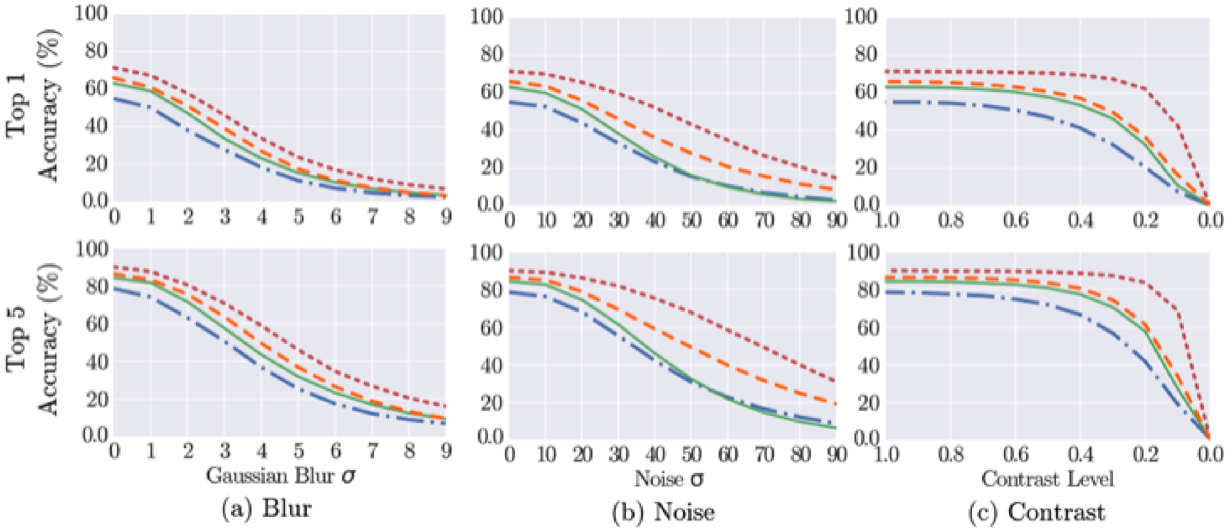
Alternate Data Augmentation A number of different types of data augmentation can be used to increase the size of the training set. This helps to improve the test results by adding more samples for the network to learn from. This is an image preprocessing technique to increase variability by adding diversity in the appearance of each mitosis, as well as the background of the mitotic region. This is intended to make the model more robust towards new examples in the test set with potentially similar characteristics to the augmentations. Ideally, we would have the ability to add other *real-life* training examples. However, augmentation is a quick and easy preprocessing step.

The optimal combination of augmentation techniques was chosen before testing the different models and parameters. The type tested include none, blur, noise, rotation, mosaic, brightness, and exposure and each will be compared to the use of no augmentation applied. In each case a new training image was added to the dataset for each image but with the applied augmentation and the unfiltered image was still included.

Blur Adding a random Gaussian blur to training images helps the trained model become more resilient. For example, certain situations could cause scanner focus changes or imaging movement effects to occur in the test set. Similar to averaging neighboring pixels, the Gaussian blur filter creates a smoothing effect reducing the amount of detail in an image.

The kernel is represented with the Gaussian bell shaped bump as described in 3.1. A three-pixel standard deviation (represented by σ in Equation 3.1) blur will be used for these tests, so up to this number of pixels may smoothed in the x or y direction. The amount will be randomized so that each individual image in the training set received anywhere from $(0,3)$ amount of blur, sampled in a uniform random manner. Using a variety of different levels of blur helps the model to be more robust. An example of the effect is shown in the images in Figure 13.

Dodge and Karam have found that blur and noise had the highest impact on simple classification tasks from a variety of CNNs when five different common quality distortions were tested: blur, noise, contrast, JPEG, and JPEG2000 compression [37]. The graphs below show the extent to which blur and noise in an image help when four state-of-the-art deep neural network models are used for image classification. Undoubtedly, image degradation is one of the most common and important real-life cases, so it is a key augmentation type to test.



(a) Blur (b) Noise (c) Contrast

Figure 11: Research Showing the Effect of Different Augmentation Techniques [37]

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{3.1}$$

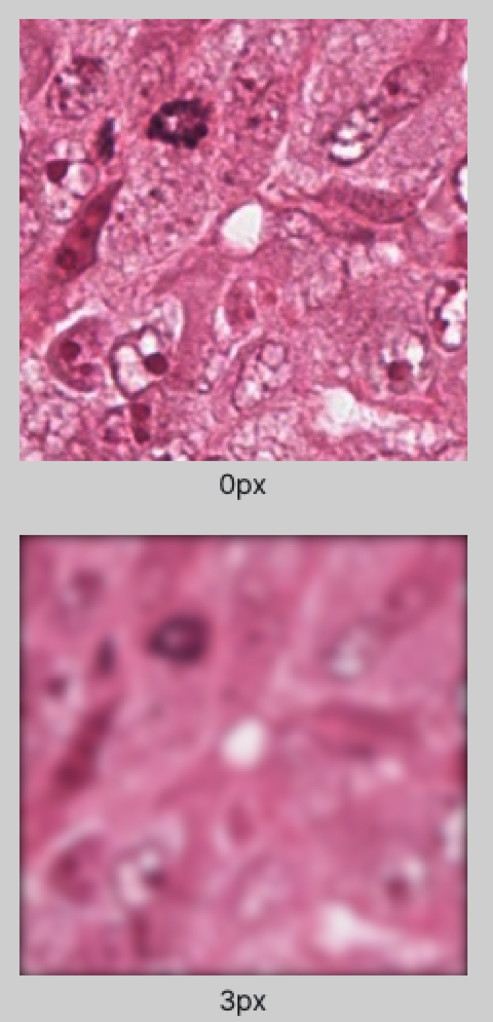


Figure 12: An Example of a 3 Pixel Standard Deviation Blur

NOISE Noise helps the model become more resilient to camera artifacts. Up to five percent noise was added for each noise augmented training image, where the percentage is the proportion of the number of pixels to be replaced. It varied in a uniform distribution, so that each image may have a different amount of noise. In this case, these new noise pixels were randomly input as either fully white or fully black. There were two additional noisy images added per original training example.

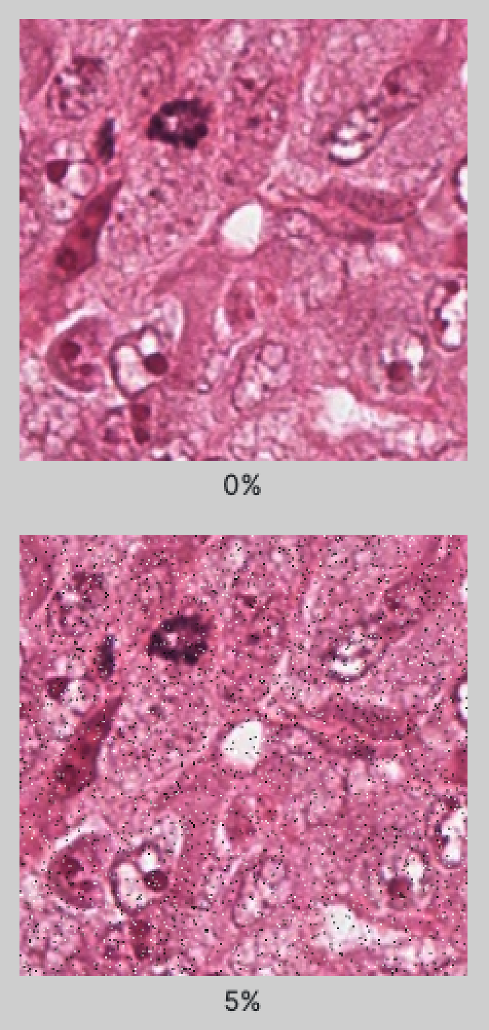


Figure 13: An Example of an Image with 5% Noise

Rotation Adding image rotations helps the model be insensitive to camera orientation. To calculate each new coordinate for each pixel in the image, the RGB (Red, Green, Blue) values for each pixel is written in the new location.

For example, in order to rotate the image coordinate by an angle θ we apply the following to the original x and the y location coordinates in the image frame of reference:

$$x^* = x\cos\theta + y\sin\theta$$

$$y^* = -x\sin\theta + y\cos\theta$$

Where x^* and y^* are the new coordinates of the same pixel in the new image. A 90 degree rotation will be used for these tests as shown in the Figure 14. For each training example, two new images will be created, with one rotated in each direction: clockwise and counter-clockwise.

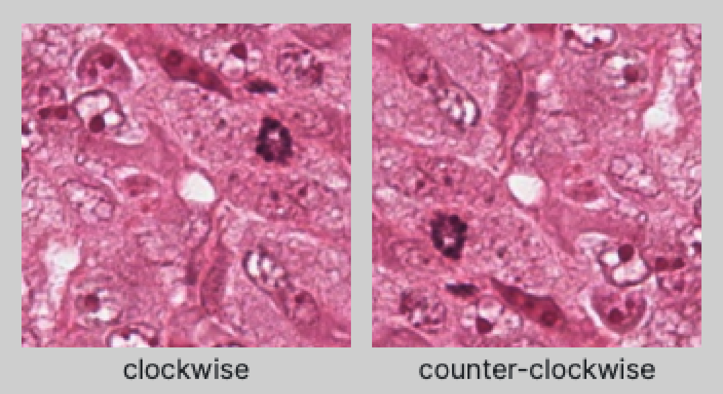


Figure 14: An Example of a 90 Degree Rotation

MOSAIC Mosaic Data Augmentation [14] helps the model perform better on small objects (such as the mitosis are in the HPF slides) by bringing them to the forefront of the image.

It randomly crops four images and combines the cropped pieces into one image, while maintaining scale. This way, slide variations that occur in one image and not another may now be combined in the newly created image. It may also help to create slides with more mitotic figures than average, which also helps the network learn new diverse examples.

BRIGHTNESS Brightness adds variability to training images to help the model become more resilient to lighting changes and scanner or microscope setting changes. A 25% brightness adjustment will be added to each new image for these tests as shown in the Figure 15.

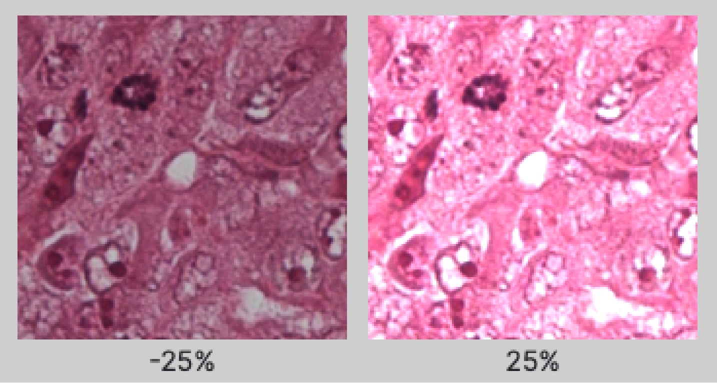


Figure 15: An Example of a 25% Brightness Adjustment

EXPOSURE Similar to brightness, exposure adds variability to training images to help the model become more resilient to lighting changes and scanner or microscope setting changes. A positive or negative 25% exposure adjustment was added to each new image for these tests as shown in the Figure 16.

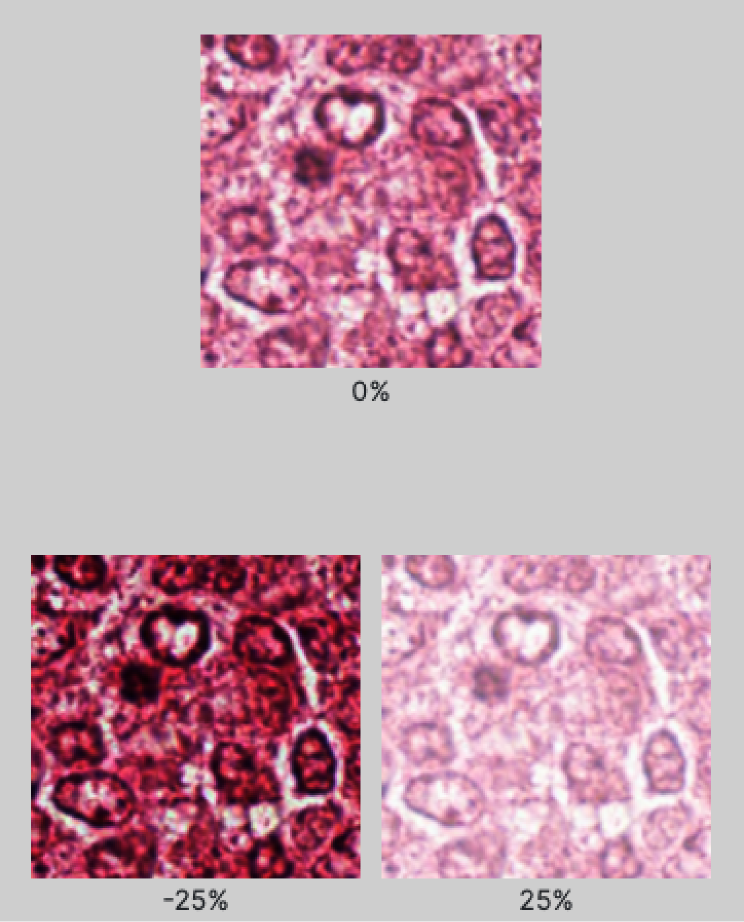


Figure 16: An Example of a $+/-25\%$ Exposure Adjustment

DECREASING THE SLIDING WINDOW FOR HPF CROPPING As described in 3.7.1, the original generated training images must be cropped from the HPF given due to the large size of an HPF and small size of the window of localization used by the network. Another way that more images can be added to the training set is to reduce the sliding window. This was done by generating similar images to the original images cropped from the HPF by cropping them slightly more in one direction or the other, slightly modifying the image frame. Decreasing the sliding window meant more examples of the same mitosis except it was captured from another location in the original image.

To calculate each new pixel for each coordinate in the full-size image, the RGB values for each pixel can be obtained from the new location.

For example, in order to shift the cropped image coordinate by ten pixels to the right and ten pixels downward we apply the following to the original x and the y location coordinates in the image frame of reference:

$$x^* = x + 10$$

$$y^* = y + 10$$

Where x^* and y^* are the new (shifted) coordinates of the same pixel in the new image. Then, the following decrements are applied to the mitosis bounding box coordinates.

$$x_1^* = x_1 - 10, y_1^* = y_1 - 10$$

$$x_2^* = x_2 - 10, y_2^* = y_2 - 10$$

Since the image shifted right and downward, the bounding box is now more left and upward in the new cropped image.

3.7 Evaluation Methods

3.7.1 Datasets & Preparation

Contest Dataset The models were trained on a subset of two different datasets in the following experiments. These are open datasets from the International Conference on Pattern Recognition (ICPR) of breast cancer histopathology in 2012 [137] and in 2014. Note, that the 2014 dataset is also known as MITOS-ATYPIA-14 [136]. Since mitotic count is an important parameter for the prognosis of breast cancer and has been under discussed in the literature, these contests were developed to address this challenging issue.

The data for the contests is for mitosis detection in images stained with standard hematoxylin and eosin (H&E) dyes. The the Aperio Scanscope XT and the Hamamatsu Nanozoomer 2.0-HT slide scanners have different resolution and are used to produce RGB images. Image annotations for the coordinates of the center pixel each mitosis are made by two senior pathologists, where if one disagrees a third will give the final say.

Since the time of the contests both have been very commonly reused among the research in this area thus far, so testing with these datasets helps compare the results to other published works. Nearly all groups who have done research in mitosis counting with machine learning this have used these datasets.

ICPR 2014 The ICPR 2014 MITOS-ATYPIA-14 contest dataset is an extension of the ICPR 2012 contest dataset and is generally more challenging and contains more images [83]. The X40 resolution training set provided consists of 2,400 labeled 1539 × 1376 pixel HPF frames, containing a total of 749 labeled mitotic cells. Half of the slides containing 749 mitosis are labeled the Apreio scanner and half are from the Hamamatsu with 753 mitosis are labeled.

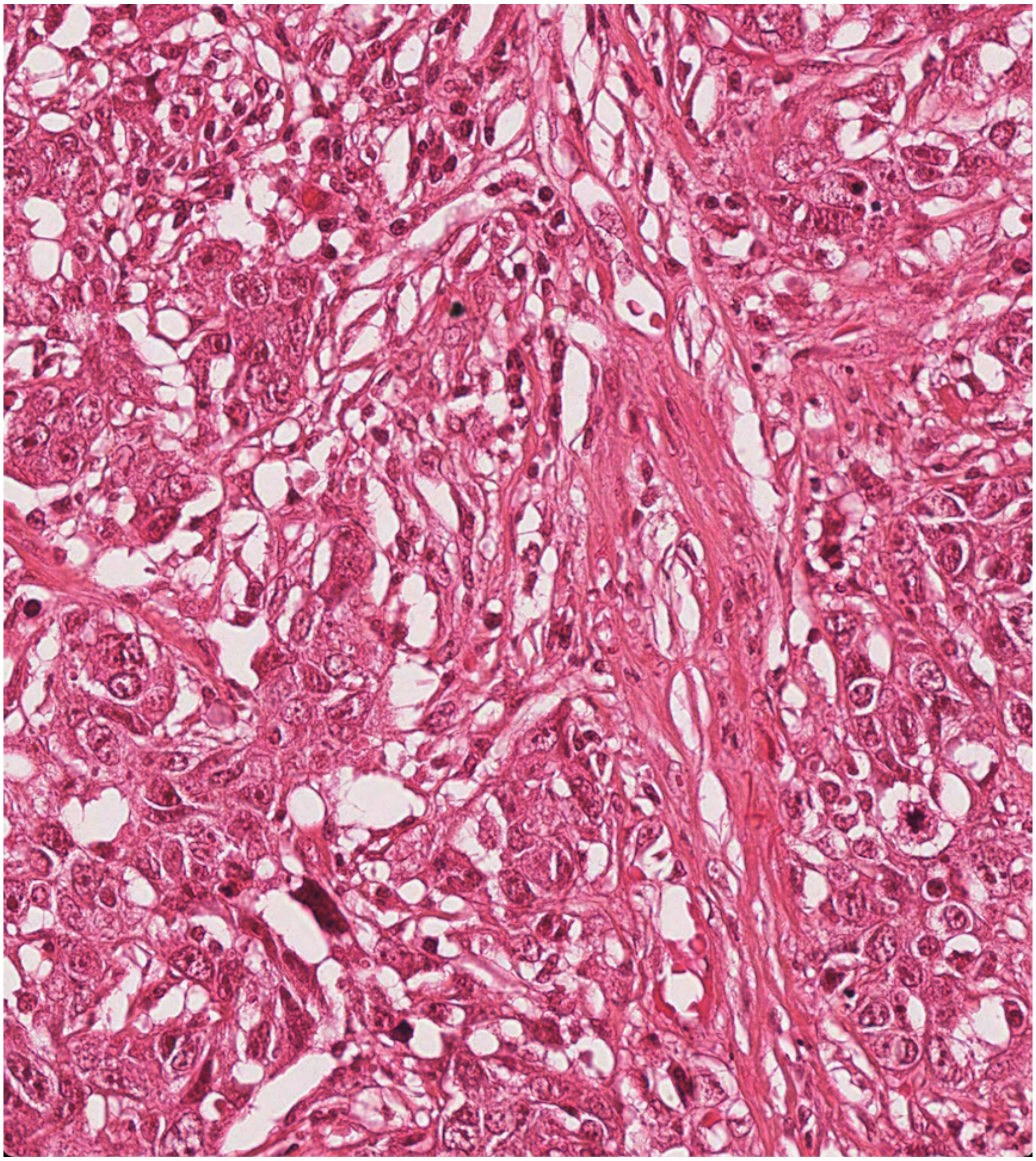


Figure 17: Full Image from ICPR 2014 [136]

ICPR 2012 The ICPR 2012 MITOSIS dataset consists 50 RGB images, where 35 images are reserved for training and 15 are for testing. Ten high-power fields (HPFs) of size 512 × 512 square micrometers at 40 × magnification are obtained from breast biopsies. The resolution of each image is 2084 × 2084 pixel image, and there are 226 and 101 mitotic cells in the training and testing sets, respectively.

The Figure 18 and Figure 19 show an example of one of the full Aperio scanner images in the dataset. Figure 18 is the same as Figure 19 except the centroid of each mitosis are highlighted in yellow.

Figure 20 and Figure 21 present the examples of the cropped regions of these two mitotic figures (in figures 18 and 19) that were used for training the network. The black lines represent the locations of the bounding box corners that were used for annotating the dataset.

Note, that the coordinates of the locations were extracted, but the black lines themselves were not added to the actual training images. (They were simply added for the purpose visualizing the bounding box here.)

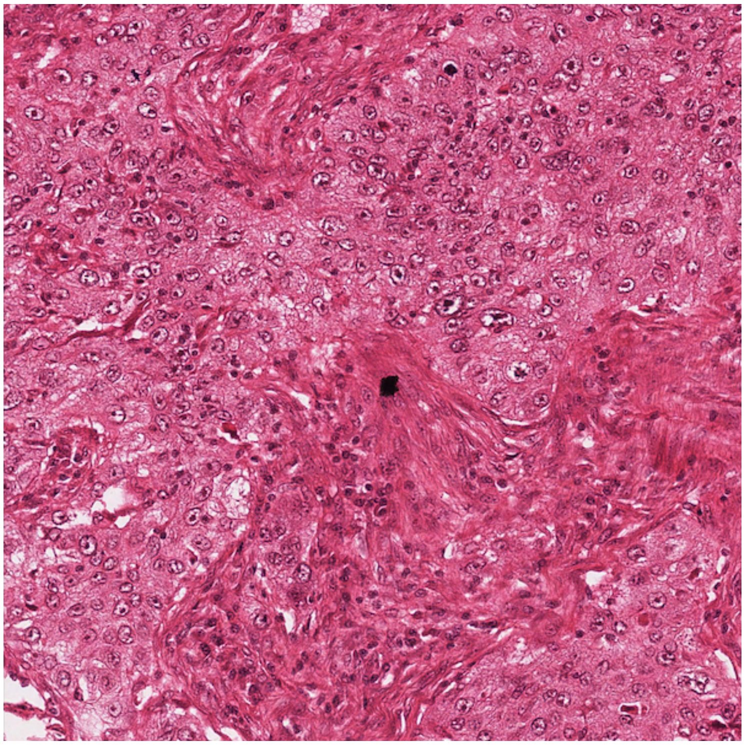


Figure 18: Full Image from ICPR 2012

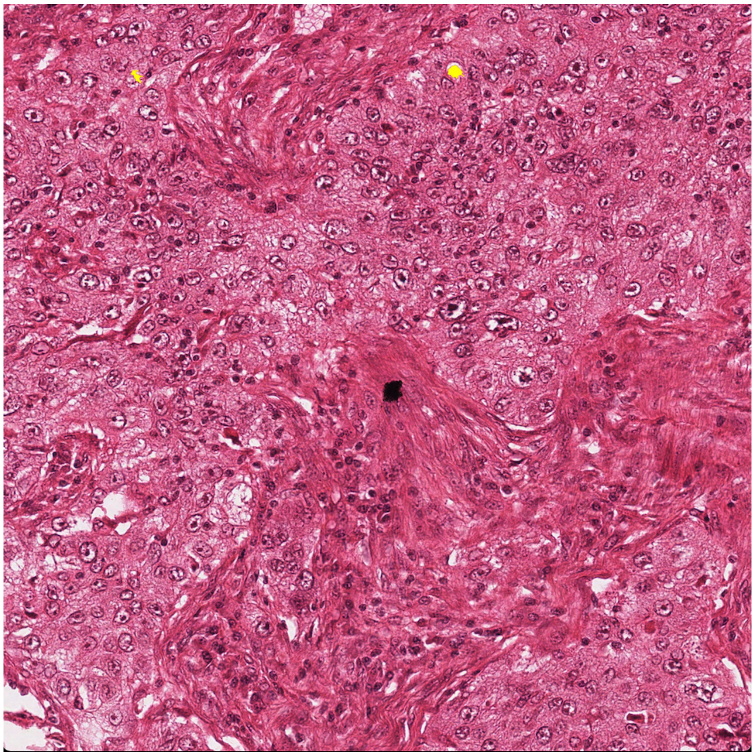


Figure 19: Full Annotated Image from ICPR 2012

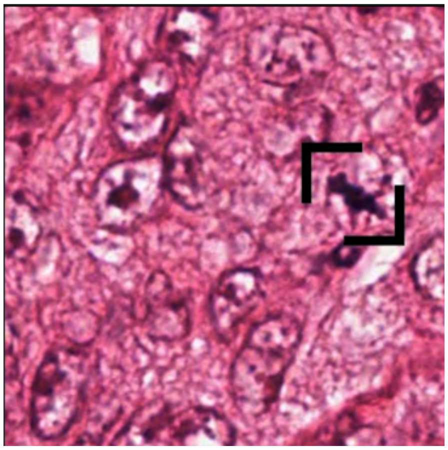


Figure 20: Cropped Region from ICPR 2012 with Framed Mitosis on Bounding Box

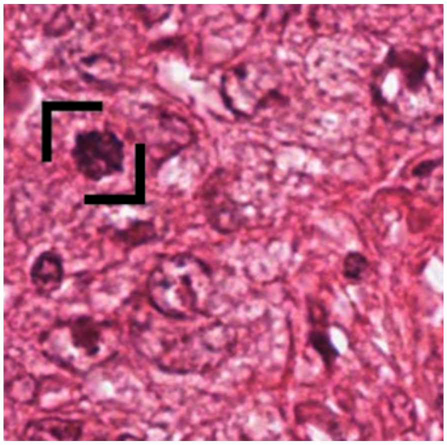


Figure 21: Cropped Region from ICPR 2012 with Framed Mitosis on Bounding Box

Train/Test Data Preparation

ICPR 2014 Preprocessing The HPF slides from the ICPR 2014 training set were first converted from the original TIFF image format to JPEG, before being cropped into 64 equal sized subsections (eg. 8×8 sections). This was done so that they can be expanded to a size large enough to display the mitosis within the size of the smallest window predicted by the network. The 2014 images are referred to as "weakly annotated" meaning that the coordinates of the bounding boxes must be derived from the centroids rather than the surface area of the mitosis, since these are the only two location points provided by the contest. The mitosis bounding box coordinates were extracted from the CSV file included in the training data by adding 25 pixels in the upper left and lower right directions from the derived centroid coordinates.

ICPR 2012 PREPROCESSING The ICPR 2012 dataset was first tested with the similar preprocessing methods to the ICPR 2014 dataset. Images were cropped into 64 (8×8) equally sized subsections for initial testing with and without the decrements in the sliding window. Then, the number of cropped regions was increased to 256 (16×16).

There are a few differences between the ICPR 2012 dataset and the ICPR 2014 dataset besides the actual data. Two are the number and the size of the images, as described above. The other main difference is that the ICPR 2012 dataset is annotated so that each of the pixels of the mitosis are given (i.e. *strongly annotated*). This allows the centroid to be derived from the center point of those pixels in order to better annotate the mitosis location. The bounding box size and shape can also be more accurately expressed from the surface area of the mitosis. This helps generate better annotations because the network can learn from a more precisely localized object, with the correct size bounding box coordinates. The images from the ICPR 2012 dataset were cropped slightly smaller than the ICPR 2014 images in order fit the entire image within the cut out locations without removing part of the image.

TEST AND TRAIN SET EXTRACTIONS The annotations of official test set for the ICPR contests are unavailable to the public so part of this training set will was used for the test set. This is similar to what most other research groups who worked with this dataset (referenced in **??**) have done, in order to be able to check the correctness of predictions by their developed framework. Here, the test set was selected randomly by extracting a set of images containing approximately the average number of mitotic cells in one slide.

The training set also needs to be significantly downsized (by removing some of the non-mitotic regions) for two main reasons. One is that in order to carry out most of the tests in the results section, the length of time for the required number of epochs is longer than the available runtime in Google Colab cloud GPU where the tests were run. Second, is due to the low density of mitotic cells in each image, image sampling is required, which is also similar to what other groups have done to solve the issue of class imbalance. The full set of images is over 100k after being split into subsections, if all non-mitotic regions are included, so for most of the tests 3k images were drawn which include all mitotic regions.

In order to verify this is comparable, one test was run for as long as possible with 50k training images with a randomly selected test set from these. However, for the main task of finding the optimal training model, version, data augmentation types, parameters, and number of epochs the train and test subsets will be used for initial tests below. The train/valid/test distribution for each will be 70%/20%/10% of the images respectively for each dataset.

MULTI-STREAM FASTER-RCNN ADDITIONAL PREPROCESSING Each of the image subsections were expanded to 1.7 times the original image size to increase the mitosis to be large enough to be compared to the smallest detectable object size of Faster RCNN. The bounding boxes for the mitosis were set by subtracting and adding 25 pixels on each the x and the y direction from the centroid multiplied by 1.7. The bounding box coordinates are initially saved as

$image - namex_1, y_1, x_2, y_2$ where x_1 and y_1 are the upper left coordinates and x_2 and y_2 are the lower right coordinates.

If we denote c_x, y to be the centroid coordinates provided by the contest administrators, then the following steps were used to generate bounding box coordinates for the image annotations:

- $x_1 = (c_x - 25) \cdot 1.7$
- $y_1 = (c_y - 25) \cdot 1.7$
- $x_2 = (c_x + 25) \cdot 1.7$
- $y_2 = (c_y + 25) \cdot 1.7$

UNET DATA AND SET-UP This image segmentation network requires ground truth images for training. In other words, a black image with a white mask in the area of localization needs to be fed to the network, simultaneously, with each RGB (red, green, blue color scale) image subsection, as shown in Figure 10 labeled part 1.

Ground truth data for the segmentation network was created for each image subsection in the training set by drawing a square of black pixels the same size as each image subsection, with a white circle of pixels in the same location as the mitosis using the annotated coordinates of the mitosis location (provided with the dataset.) Images input into UNet must also were slightly downsized to avoid padding issues.

UNet was then trained for 80 iterations on half of the Faster R-CNN training set consisting of only the Aperio scanner images and their corresponding ground-truth images. Then, it was tested on the images from both scanners to create the full set of segmented training data for the segmentation stream input of Faster R-CNN.

It is important that at least half of the test data for UNet is not in the training set since the actual test images for the full multi-stream network was not in the training set of UNet. This avoids overfitting UNet to allow for imperfections in segmented training data.

YOLO AUGMENTATIONS AND ADDITIONAL PREPROCESSING The input to the YOLO implementation requires images to be resized to 416×416 or 448×448 and image bounding box annotations to be scaled by height and width. The first two coordinates are the centroid and the second two coordinates also need to be stated as width and height. So the new second coordinates were modified from the training set used for Faster-RCNN and calculated by subtracting $x_2 - x_1$ and $y_2 - y_1$ and the first by adding half of that to the original, then they are divided by the height and width of the image.

The following calculation were used to generate the new coordinates for x and y:

- $x = (x_1 + (x_2 - x_1) \cdot 1/2) \cdot 1/w$
- $y = (y_1 + (y_2 - y_1) \cdot 1/2) \cdot 1/h$

The following preprocessing was applied to each train and test image described above and each of the augmentations were again scaled and oriented accordingly.

- Resize to 416×416 (Stretch)

DATA AUGMENTATION Data augmentation was applied to the training set used for the comparison of the different versions of YOLO. The following was applied to create a total of three versions of each source image

- Random exposure adjustment of between -25 and +25 percent

An example of the data with augmentation applied a couple of images in the initial training batch is shown below. Note, this also includes augmentation applied by YOLOR itself, internally (eg. Mosaic).

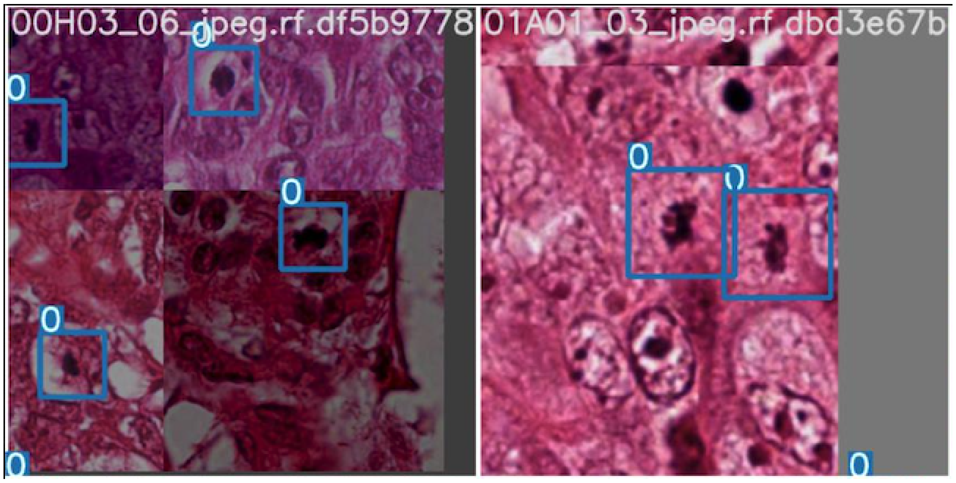


Figure 22: Example of the Data Augmentation to the Training Batch

3.7.2 ACCURACY CALCULATION

According to the contest evaluation criteria, a correct detection is the one that lies within 32 pixels from this centroid of the ground truth mitosis. The score for the test here was calculated in the same way as the contestants are ranked according to their F-score. This is a harmonic mean of precision and recall (sensitivity), as described below.

$$F-score = \frac{2 \cdot (precision \cdot sensitivity)}{(precision + sensitivity)} \tag{3.2}$$

The precision measures how accurate the predictions are using the percentage of the correct predictions out of the total. It is calculated using the FP which represents the number of false positive predictions, and TP which is the number of true positive predictions, as shown below:

$$Precision = \frac{TP}{FP+TP} \tag{3.3}$$

The recall measures how well all the positives are found in the test set, where FN is the number of false negatives (those ground truths which were not detected), as shown below

$$Recall = \frac{TP}{FN+TP} \tag{3.4}$$

3.7.3 Software & Hardware

The Google Colab cloud resource was used for the GPU access. The architecture is limited to NVIDIA P100 or T4, with RAM to 25 GB. This GPU runtime can stay connected for up to 12 hours, while Colab Pro may stay connected for up to 24 hours. However, Google Colab GPU session durations are not guaranteed. The timeouts are relatively frequent when the session is left idle.

YOLOv5 The YOLOv5 implementation that was used in the following tests is an open-source software written in the Ultralytics framework [71]. YOLOv5 is an unofficial implementation created based on YOLOv4 by Glenn Jocher in mid-2020. Also note that Jocher was the one who was credited with adding a new type of data augmentation called Mosaic Data Augmentation, which is included in the implementation.

The family of object detection architectures and models is pretrained on the COCO dataset [92] and implemented in PyTorch (not Darknet as most of the previous YOLO versions were). The implementation has been proven to far excel in speed in accuracy over EfficientDet [153], as shown in Figure 23.

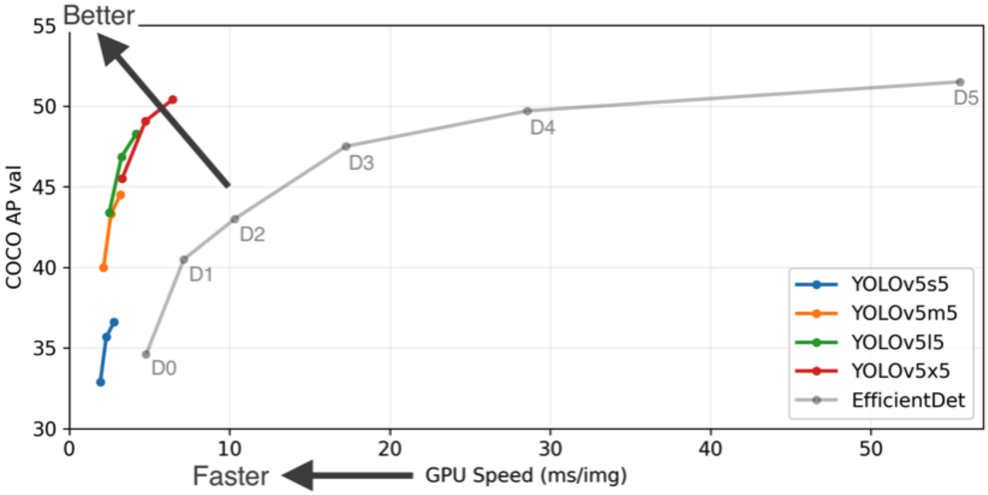


Figure 23: Comparison of YOLOv5p5 Models and EfficientDet [14].

The layers and parameters of the backbone and head of YOLOv5 are shown in Table 19 and Table 20, respectively. The different model configurations can be found under the **model** folder in the repository.

The version p6 of each scale includes a larger head and backbone consisting of additional layers, as well as an

From	Number	Module	Args
-1	1	Focus	[64, 3]
-1	1	Conv	[128, 3, 2]
-1	3	C3	[128]
-1	1	Conv	[256, 3, 2]
-1	9	C3	[256]
-1	1	Conv	[512, 3, 2]
-1	9	C3	[512]
-1	1	Conv	[1024, 3, 2]
-1	1	SPP	[1024, [5, 9, 13]]
-1	3	C3	[1024, False]

Table 19: YOLOv5 Backbone [71]

From	Number	module	Args
-1	1	Conv	[512, 1, 1]
-1	1	nn.Upsample	[None, 2, 'nearest']
[-1, 6]	1	Concat	[1]
-1	3	C3	[512, False]
-1	1	Conv	[256, 1, 1]
-1	1	nn.Upsample	[None, 2, 'nearest']
[-1, 4]	1	Concat	[1]
-1	3	C3	[256, False]
-1	1	Conv	[256, 3, 2]
[-1, 14]	1	Concat	[1]
-1	3	C3	[512, False]
-1	1	Conv	[512, 3, 2]
[-1, 10]	1	Concat	[1]
-1	3	C3	[1024, False]
[17, 20, 23]	1	Detect	[nc, anchors]

Table 20: YOLOv5 Head [71]

19,27	44,40	38,94
96,68	86,152	180,137
140,301	303,264	238,542
436,615	739,380	925,792

Table 21: YOLOv5-p6 Anchors [71]

Model	Width Multiple	Depth Multiple
YOLOv5s	0.5	0.33
YOLOv5m	0.75	0.67
YOLOv5l	1.0	1.0
YOLOv5x	1.25	1.33

Table 22: YOLOv5 Model Versions

additional anchor scale. The larger backbone has seven additional convolutional layers in the head, and two additional convolutional layers in the backbone. As shown in 24, this helps it excel over the standard YOLOv5 on the COCO [92].

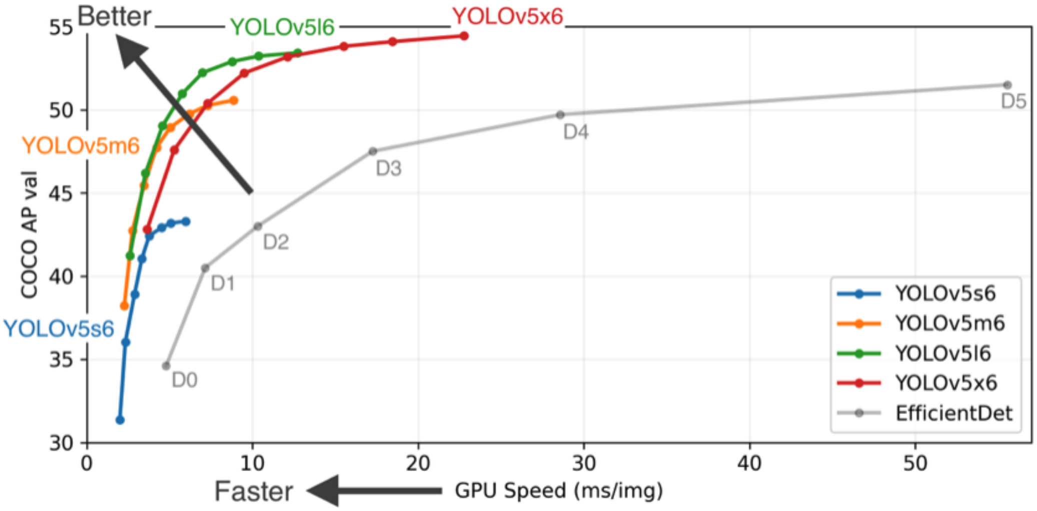


Figure 24: Comparison of YOLOv5p5 Models and EfficientDet [14]

The p-5 anchor boxes are the same as YOLOv3 as shown in Table 25, but the p-6 anchor boxes are slightly different and include a larger size as an additional box, as shown in Table 21.

The different model scales are shown in Table 22.

12,16	19,36	40,28
36,75	76,55	72,146
142,110	192,243	459,401

Table 23: YOLOv4-Scaled Anchors [71]

YOLOv4-Scaled The official implementation of YOLOv4-Scaled [166] is on Github, and it makes use of the Pytorch framework. This includes a small version (for CPU), a standard version, and a large version (for cloud GPU), as well as versions that support 2 or more GPUs. In this case yolov4-csp from the yolov4-large branch for cloud GPU was used. The different model configurations can be found under the **model** folder in the repository.

YOLOv4-scaled has almost the same backbone as YOLOv3 shown in Table 26 except for the **Bottleneck** module becomes **BottleneckCSP**. The head of the model is shown in Table 24 and the default anchor boxes will be used as shown in 23.

YOLOv3 The Github repository for the YOLOv3 implementation used is an open-source software written in the Ultralytics framework [71]. The primary implementation is the YOLOv5 [72], but the repository also contains the model parameters and layers for the YOLOv3 network.

The backbone and head is shown in Table 26 and Table 27, respectively. The head and backbone are significantly different from YOLOv5. The default anchor boxes will be used as shown in 25, which also differ from the other versions.

The different model configurations can be found under the **model** folder in the repository.

YOLOR The official implementation of YOLOR [170] is on Github. This includes a few different default or preset configurations. The **yolor-p6.cfg** was used in this case. The different model configurations can be found under the **config** folder in the repository.

3.8 Results

The author carried out the following experiments as described above and recorded the F-score (as well as the precision and recall in most cases).

3.8.1 Multi-stream Faster RCNN Results

The Multi-stream Faster RCNN model achieved an F-measure of 0.507. This is already significantly higher than contest winners score of 0.356, and the network was trained on a smaller portion of the training set with little data

From	Number	module	Args
-1	1	SPPCSP	[512]
-1	1, Conv	[256, 1, 1]	
-1	1	nn.Upsample	[None, 2, ’nearest’]
8	1	Conv	[256, 1, 1]
[-1, -2]	1	Concat	[1]
-1	2	BottleneckCSP2	[256]
-1	1	Conv	[128, 1, 1]
-1	1	nn.Upsample	[None, 2, ’nearest’]
6	1	Conv	[128, 1, 1]
[-1, -2]	1	Concat	[1]
-1	2	BottleneckCSP2	[128]
-1	1	Conv	[256, 3, 1]
-2	1	Conv	[256, 3, 2]
[-1, 16]	1	Concat	[1]
-1	2	BottleneckCSP2	[256]
-1	1	Conv	[512, 3, 1]
-2	1	Conv	[512, 3, 2]
[-1, 11]	1	Concat	[1]
-1	2	BottleneckCSP2	[512]
-1	1	Conv	[1024, 3, 1]
[22,26,30]	1	Detect	[nc, anchors]

Table 24: YOLOv4-Scaled Head [71]

10,13	16,30	33,23
30,61	62,45	59,119
116,90	156,198	373,326

Table 25: YOLOv3 Anchors [71]

From	Number	module	Args
-1	1	Conv	[32, 3, 1]
-1	1	Conv	[64, 3, 2]
-1	1	Bottleneck	[64]
-1	1	Conv	[128, 3, 2]
-1	2	Bottleneck	[128]
-1	1	Conv	[256, 3, 2]
-1	8	Bottleneck	[256]
-1	1	Conv	[512, 3, 2]
-1	8	Bottleneck	[512]
-1	1	Conv	[1024, 3, 2]
-1	4	Bottleneck	[1024]

Table 26: YOLOv3 Darknet53 Backbone [71]

From	Number	module	Args
[-1	1	Bottleneck	[1024, False]]
-1	1	Conv	[512, [1, 1]]
-1	1	Conv	[1024, 3, 1]
-1	1	Conv	[512, 1, 1]
-1	1	Conv	[1024, 3, 1]
-2	1	Conv	[256, 1, 1]
-1	1	nn.Upsample	[None, 2, ’nearest’]]
[-1, 8]	1	Concat	[1]
-1	1	Bottleneck	[512, False]
-1	1	Bottleneck	[512, False]
-1	1	Conv	[256, 1, 1]
-1	1	Conv	[512, 3, 1]
-2	1	Conv	[128, 1, 1]
-1	1	nn.Upsample	[None, 2, ’nearest’]
[-1, 6]	1	Concat	[1]
-1	1	Bottleneck	[256, False]
-1	2	Bottleneck	[256, False]
[27, 22, 15]	1	Detect	[nc, anchors]

Table 27: YOLOv3 Head [71]

Augmentation Type	F-score
None	0.77
Noise	0.94
Blur	0.94
Rotation	0.93
Blur & Rotation	0.94
Mosaic	0.94
Brightness	0.94
Exposure	0.94

Table 28: Tests with YOLOR with Different Data Augmentation Techniques Applied with ICPR 2014

augmentation, besides resizing the images. This is also higher than the recent similar high scores with similar tests on this dataset using deep learning, achieving scores of 0.437 and 0.442 [84].

3.8.2 YOLO Results

The YOLO version/model, and combined training and testing runtimes (in hours), are shown in the columns of the Tables below.

Results of Data Augmentation The Table 29 below shows how the different augmentation techniques applied to the dataset compared to the test without augmentation. The *Augmentation Type* column is the type of augmentation applied described above.

For each of the tests, the YOLOR model was trained for 60 epochs and a batch size of 8. There is a significant increase in the score for the dataset with image augmentation, however, it takes nearly twice as long to train for the same number of epochs due to the increase in the size of the dataset. Therefore, for further testing only one augmentation technique will be applied to the datasets since the combination of multiple augmentations did not significantly effect the results, besides increasing the training time.

Further, when no augmentation was applied and the training time was increased to the same amount as all of the other tests (the number of epochs were doubled), the F-score was still not as high as it was with augmentation; it only increased to 0.87 and 0.86 for ICPR 2012 and ICPR 2014, respectively. The training time for ICPR 2012 was around 0.32 hours for each test with augmented data and 60 epochs. The training time was around 1.4 hours for ICPR 2014, for the same.

YOLOv5 Each preset model scale and size of YOLOv5 was tested with a batch size of 16 for both the p5 and p6 models. They were chosen to be trained for 120 epochs because the F-score stopped increasing at slightly over 100

Augmentation Type	F-score
None	0.67
Noise	0.92
Blur	0.92
Rotation	0.93
Blur & Rotation	0.94
Mosaic	0.92
Brightness	0.94
Exposure	0.96

Table 29: Tests with YOLOR with Different Data Augmentation Techniques Applied with ICPR 2012

Model	Time (hrs)	Precision	Recall	F-score
YOLOv5s	1.52	0.93	0.95	0.94
YOLOv5m	1.53	0.95	0.95	0.95
YOLOv5l	2.29	0.90	0.94	0.92
YOLOv5x	4.09	0.95	0.94	0.94

Table 30: Tests with YOLOv5-p5 for 120 Epochs with ICPR 2014

epochs. Therefore, this is likely the highest feasible F-score with this setup. The resulting scores for the p5 model on the test set with each of the different model scales are shown in Tables 30 and 31 for ICPR 2014 and 2012 datasets, respectively.

Overall the version *m* consistently had a slightly higher F-score, but the version did not make a large difference in the results. For ICPR 2014 and 2012, F-scores of up to 0.95 and 0.96, respectively, were achieved using YOLOv5m with the augmented training data when trained for 120 epochs. However, the version *l* and *x* required far longer runtime without an increase in F-score for both datasets.

The F-scores for the different model scales for the version p6 are shown in Tables 32 and 33 for ICPR 2014 and

Model	Time (hrs)	Precision	Recall	F-score
YOLOv5s	0.18	0.9	1.0	0.94
YOLOv5m	0.26	0.96	0.96	0.96
YOLOv5l	0.50	0.96	0.96	0.96
YOLOv5x	0.79	0.92	0.96	0.94

Table 31: Tests with YOLOv5-p5 for 120 Epochs with ICPR 2012

Model	Time (hrs)	Precision	Recall	F-score
YOLOv5s6	1.78	0.91	0.91	0.91
YOLOv5m6	3.16	0.89	0.82	0.85
YOLOv5l6	10.70	0.90	0.81	0.85
YOLOv5x6	20.06	-	-	-

Table 32: Tests with YOLOv5-p6 for 120 Epochs with ICPR 2014

Model	Time (hrs)	Precision	Recall	F-score
YOLOv5s6	0.19	0.96	1.0	0.98
YOLOv5m6	0.37	0.79	1.0	0.88
YOLOv5l6	0.58	0.79	1.0	0.88
YOLOv5x6	1.01	1.0	0.87	0.93

Table 33: Tests with YOLOv5-p6 for 120 Epochs with ICPR 2012

2012 datasets, respectively. The versions *l* and *m* required far longer runtime with a decrease in F-score from the *s* version for both datasets. The highest F-score and lowest runtime was obtained with *s* for both datasets.

Results for ICPR 2014 could not be obtained for YOLOv5x6 because the cloud runtime disconnected after 20 hours. However, for the ICPR 2012 dataset the version *x* was useful and actually resulted in a higher F-score than the *l* and *m* versions.

On the other hand, overall for both datasets the F-score also was lower when the version p6 was used than when the standard p5 version was used. The p5 version had nearly half the runtime length for training.

Table 34 lists the Precision, Recall, and F-score for the test set when trained with version m-p5 at lower numbers of epochs. This was the version shown above with the highest F-score in comparison to the rest of them with 120 epochs. Therefore, it was of interest to see if the verify 120 epochs were required for training since more epochs leads to longer runtimes. It may also be helpful to see how much the score changes with lower runtimes.

Even at 60 epochs, an F-score of 0.89 was obtained with 46 minutes of training time for ICPR 2014. An F-score of 0.92 was also obtained in only around 10 minutes for ICPR 2012.

Table 36 lists the Precision, Recall, and F-score for the test set when trained with version s-p6 at lower numbers of epochs with ICPR 2012. The *s* version received the highest F-score for yolov5-p6, for this dataset so further testing was done to see if less epochs could result in both a lower runtime with a comparable score. The runtime of training was in fact shorter, however, the F-score was still lower than the p5 version for any number of epochs.

Epochs	Time (hrs)	Precision	Recall	F-score
15	0.70	0.89	0.67	0.77
60	0.76	0.89	0.88	0.89
90	0.95	0.91	0.92	0.92

Table 34: Tests with YOLOv5m-p5 with ICPR 2014

Epochs	Time (hrs)	Precision	Recall	F-score
15	0.05	-	-	-
60	0.18	0.82	1.00	0.92
90	0.26	0.96	0.96	0.96

Table 35: Tests with YOLOv5m-p5 with ICPR 2012

Epochs	Time (hrs)	Precision	Recall	F-score
15	0.03	0.0	0.0	0.0
60	0.10	1.00	0.71	0.84
90	0.15	0.92	0.85	0.88

Table 36: Tests with YOLOv5s-p6 with ICPR 2012

Epochs	Time (hrs)	Precision	Recall	F-score
20	0.49	0.90	0.84	0.87
40	0.95	0.89	0.93	0.91
60	1.44	0.90	0.95	0.93
80	1.91	0.89	0.95	0.92

Table 37: Tests with YOLOv4-Scaled with ICPR 2014

Epochs	Time (hrs)	Precision	Recall	F-score
20	1.35	0.91	0.77	0.84
60	1.63	0.90	0.92	0.91
120	2.69	0.90	0.92	0.91

Table 38: Tests with YOLOv3 with ICPR 2014

YOLOv4-Scaled Table 37 shows the tests with the scaled YOLOv4-csp with a batch size of 16 and a range of numbers of epochs.

For the ICPR 2014 dataset it takes around 60 epochs for the F-score to reach its highest F-score of around 0.93. The runtime was approximately the same as YOLOv5m for a similar F-score so scaling did not improve results.

YOLOv3 Table 38 shows the tests with the scaled YOLOv3 model with a batch size of 16 and a range of numbers of epochs for ICPR 2014. It takes around 60 epochs for the F-score to reach its highest F-score of around 0.91.

Table 39 shows the tests with the scaled YOLOv3 model with a batch size of 16 and a range of numbers of epochs for ICPR 2012. It takes around 60 epochs for the F-score to reach its highest F-score of around 0.92.

However, although the F-score is not that much lower, the runtime is much longer than the YOLOv5 and YOLOv4-scaled models for both datasets.

Epochs	Time (hrs)	Precision	Recall	F-score
20	0.09	0.82	1.0	0.90
60	0.25	0.82	1.0	0.92
120	0.50	0.81	1.0	0.91

Table 39: Tests with YOLOv3 with ICPR 2012

Epochs	Time (hrs)	Precision	Recall	F-score
30	0.84	0.94	0.91	0.93
60	1.34	0.91	0.89	0.90
120	2.64	0.92	1.0	0.92

Table 40: Tests with YOLOR on ICPR 2014

Epochs	Time (hrs)	Precision	Recall	F-score
30	0.18	0.92	0.92	0.92
60	0.32	0.97	0.94	0.96
120	0.65	0.93	0.94	0.94

Table 41: Tests with YOLOR on ICPR 2012

Table 42: Tests with YOLOv5-p5 combined with YOLOR for 30 Epochs with ICPR 2012

Time (hrs)	Precision	Recall	F-score
0.18	0.97	0.94	0.96

YOLOR Table 40 shows the tests with YOLOR when trained with increasing numbers of epochs, using a batch size of 8. This model has lower runtime and a higher F-score for any number of epochs. Only 30 epochs are required to reach the highest F-score of 0.93.

Table 41 shows the tests with YOLOR when trained with increasing numbers of epochs, using a batch size of 8 for ICPR 2012. On this dataset this model provides a similar runtime and F-score for any number of epochs to YOLOv5.

COMBINING THE PREDICTIONS OF THE YOLOV5M-P5 MODEL WITH YOLOR MODEL The YOLOv5m-p5 model with YOLOR model run in parallel on separate GPUs at the same time, for only 30 epochs. Since the predictions made with YOLOv5m-p5 and with YOLOR were both very high yet both had different predictions, the combination of predictions was used to produce consistently the highest final scores and lowest runtimes (over multiple tests), as shown in Table 42. For example, YOLOv5m-p5 helped to eliminate false positives predicted by YOLOR, resulting in a higher precision than YOLOR alone.

COMBINING BOTH DATASETS Tables 43 and 49 below show the results of combining the ICPR 2012 and ICPR 2014 mitosis counting training datasets. For each test, YOLOR was trained for both 60 and 120 epochs with a batch

Model	Epochs	Time (hrs)	Precision	Recall	F-score
YOLOR	60	0.44	0.93	0.96	0.94
YOLOR	120	0.87	0.90	0.98	0.94

Table 43: Tests with Combined Training Sets and YOLOR on ICPR 2014 Test Set

Model	Epochs	Time (hrs)	Precision	Recall	F-score
YOLOvR	60	1.5	0.93	0.96	0.92
YOLOvR	120	3.2	0.89	1.0	0.96

Table 44: Tests with Combined Training Sets and YOLOR on ICPR 2012 Test Set

size of 8. The test set in 43 is extracted from the ICPR 2014 dataset and the test set from 49 is extracted from the ICPR 2012 dataset. Note, that these are same test sets were used in these tests as were used in all of the tests above to find the optimal run configuration.

By combining the training sets, the highest F-scores were obtained on both the ICPR 2012 test set and the ICPR 2014 test set.

The runtime for training the ICPR 2014 dataset with YOLOR was also the lowest with the highest F-score. Although the runtime for training was much higher, the highest F-score was obtained on the ICPR 2012 test dataset.

CHANGING THE CROPPING METHOD WITH ICPR 2012 Table 45 shows the results of the tests with larger or more cropped images from the original HPF slides, described in 3.7.1. The runtime for training was more than twice as long as in many tests with the ICPR 2012 training set and the F-score was still lower. See the 3.9 for more details on why.

ADDING THE DATA FROM ANOTHER SCANNER In all of the previous tests data from both the Aperio Scanscope XT and the Hamamatsu Nanozoomer 2.0-HT slide scanners were included in the training set. In order to test whether including the data from both of the scanners helped in prediction, a test set of images from each individual scanner

Epochs	Time (hrs)	Precision	Recall	F-score
120	3.1	0.88	0.88	0.88

Table 45: Tests with YOLOR with ICPR 2012 with Larger/More Cropped Images

Train Dataset	Test Dataset	F-score
Aperio XT & Hamamatsu Nanozoomer 2.0-HT	Aperio XT	0.94
Aperio XT	Aperio XT	0.94
Aperio XT	Hamamatsu Nanozoomer 2.0-HT	0.81
Aperio XT & Hamamatsu Nanozoomer 2.0-HT	Hamamatsu Nanozoomer 2.0-HT	0.91
Hamamatsu Nanozoomer 2.0-HT	Hamamatsu Nanozoomer 2.0-HT	0.95
Hamamatsu Nanozoomer 2.0-HT	Aperio XT	0.95

Table 46: YOLOR with Different Combinations of Train and Test Datasets ICPR 2014

was sampled and tested with a network trained with only the Aperio or only the Hamamatsu images or with both, as shown in the tables 47 and 46.

As shown in the results in the Table 46 below, adding the Hamamatsu scanner data to the training set did not help in prediction in the tests on the Aperio scanner data with the ICPR 2014 dataset. The resulting F-score did not decrease when the Hamamatsu scanner data was added to the training set when tested on the Aperio (only) scanner test set.

However, when the training set consisted of the Hamamatsu combined with the Aperio or just consisted of the Hamamatsuthe network predicted Hamamatsu scanner dataset alone, better. Interestingly, the network predicted the Hamamatsu scanner test set best when the Aperio data was removed from the training set.

Therefore, when the network was trained on both scanner data it was not able to better predict the images from the test set consisting of one scanner alone. Adding the data from another scanner to the training set also significantly increases the runtime for training.

However, it is also interesting to note that the Hamamatsu scanner *only* training set helped to predict both the Aperio test set alone and the Hamamatsu alone test set the best, but only when it was trained with the Aperio images excluded. The Hamamatsu scanner *only* training set actually produced a very slight increase in the F-score by 0.01.

For the ICPR 2012 dataset, there were not enough sample images to run the same tests. Therefore, I compared removing one scanner dataset from the test set to testing both. This will also be interesting to compare the change in F-score to the combined test set.

Next, both the Aperio and the Hamamatsu slide scanners were combined for the ICPR 2012 training dataset. The trained network predicted either the the Aperio *only* test set or the Hamamatsu slide scanner combined with the Aperio data test set at slightly different accuracy.

Therefore, the combined training set likely predicted the Hamamatsu slide scanner data at a lower accuracy than the Aperio. However, Similar to the ICPR 2014 dataset, it also may be the case that one scanner provides very slightly better training data for both slide scanners.

Train Dataset	Test Dataset	F-score
Aperio XT & Hamamatsu Nanozoomer 2.0-HT	Aperio XT	0.94
Aperio XT & Hamamatsu Nanozoomer 2.0-HT	Both	0.92

Table 47: YOLOR with Different Combinations of Train and Test Datasets ICPR2012

YOLOR Full Aperio Dataset

Nearly Unfeasible Task In order to verify that the results on the full dataset (eg. showing the network all possible image data in training) would be comparable to the results on the test set when the network is trained on the subset that was extracted, the full Aperio XT dataset was trained for as long as possible. Even this dataset contained over 76k images *without augmentation*, so this is an extremely difficult task to do on a standard GPU, *even with the fastest and most powerful training network and setup.*

Limited Testing Setup The maximum possible training time that was able to be reached in this case was for 80 epochs. Due to the runtime restrictions on Google Colab, over three days were required for the training the network on the large dataset. The trained weights were saved before night hours, and then the network was trained further starting the next day from those weights. The reason it was low for so long is because the network did not have enough time to see all of the examples of mitosis images. As described above, most researchers have augmented the data to include more mitosis due to the relatively low density in each slide.

Result When the F-score reached 0.84 at this point, it was considered to be comparable to the smaller datasets used. This is a good assumption especially considering that the score was still increasing when it was stopped and could have gone higher.

3.9 Discussion

.

In the following subsections, each portion of the results will be covered in the roughly the same order that it was covered above.

3.9.1 Multi-stream Faster RCNN

Multi-stream Faster RCNN vs. Faster RCNN The Multi-stream Faster RCNN model achieved an F-score that exceeded the basic Faster RCNN model. Using this train/test setup, it also out-performed the contestants of the

ICPR contest, and many other implementations that have been developed in research studies prior to this time in the mitosis counting contests with similar test/train set-ups.

Using the second stream with the UNet mitosis segmentation, helped to improve the F-score by adding in features of interest. This way the convolutional neural network was better able to learn new mitosis features which did not exist from the RGB image alone. The segmented images created by the output of UNet were included in the output feature map from the last layer of the CNN to take these new features into account, helping to improve overall prediction accuracy.

It is likely that these are the very slight or subtle differences in the mitosis which are hardly visible for human eye can to see. This is an extremely useful quality for possible future real life applications because the pathologist could get tired and his or her eye may not be able to see slight differences quite as well. If the mitosis happens to get initially mislabeled, the system built based on this implementation would be able to pick up the change and catch the difference in a subtle feature.

COMPARISON TO OTHER RCNN-BASED MODELS The resulting F-score obtained was 0.508 is significantly higher than contest winners score of 0.356. Further, Table 49 shows a comparison of the precision, recall, and F-score to other top-performing groups who have evaluated an RCNN-based model on the ICPR 2014 MITOSIS Dataset. Many of the top results described in literature either used a version of the regional CNN (eg. RCNN) or a deep cascaded network. The Multi-stream Faster RCNN achieved a higher F-score than a single stream alone as well as most others who have tested RCNN-based models on the public data.

Table 48: Performance Comparison on 2014 MITOSIS Dataset

Method	**Results**		
	Precision	***Recall***	***F-score***
Lightweight RCNN [88]	0.40	0.45	0.427
FRCNN	0.43	0.44	0.435
DeepMitosis (DeepDet+Seg+Ver) [85]	0.43	0.44	0.437
CasNN [22]	0.46	0.51	0.482
MS-FRCNN	0.48	0.54	0.508

FASTER-RCNN-BASED VS RCNN MODELS Since the number of objects of interest is not fixed, a standard CNN followed by a fully connected layer is inadequate because objects (or mitotic figures) have different sizes and locations. It is too computationally expensive to check all of the sizes and locations. However, these RCNN models have the ability to use region proposal algorithms to extract regions of interest using selective search, SVM, and RoI pooling techniques to choose ideal candidates rather than checking all the locations. The resulting bounding boxes can then be updated on each iteration to get closer and closer to the precise location of the ground truth. Since only the parts of the image that are likely to contain objects must be checked, this usually results in faster training times over older methods.

This is likely why many top-performing models are RCNN-based. CasNN also came closest to matching the score, using a coarse retrieval model to identify and localize mitotic candidates while preserving a high sensitivity. Unfortunately, this method is too slow to be clinically applicable, having an inference speed of 4.62 seconds per HPF.

DeepMitosis also took advantage of an RCNN with RPN while making use of segmentation features to help refine mitotic regions and obtained a high score as well. The Lightweight RCNN achieves a slightly lower score from both which may due to the fact that it uses a course candidate extractor instead of an RPN. These methods are faster though, as the lightweight RCNN takes 0.83, and DeepMitosis takes 0.72 seconds. However, Faster RCNN improves upon the RCNN by using a region proposal network that takes the feature maps as input and outputs region proposals, making it quicker. This proposed method takes only about 0.55s to test a single HPF, making it faster than RCNN-based other methods.

LACK OF HARDWARE FOR FASTER RCNN-DERIVED MODELS However, without the use of a highly efficient GPU, the amount of available data memory to work with and the frame processing rate could not reach its full potential. Additionally, there are a number of different improvements that could have been made to the actual design of Faster RCNN model and will be described in more detail in the Future Work 3.10.1.

3.9.2 YOLO VS MULTI-STREAM FASTER RCNN

Overall, using any version of YOLO with any training and testing dataset was superior in terms of runtime and prediction accuracy (F-score) versus Faster-RCNN, Multi-stream Faster RCNN, or pretty much any other model in literature.

Unlike the region proposal methods, YOLO [126] *Only Looks Once* (hence the name YOLO, which stands for *You Only Look Once*) and combines the CNN used to predict the bounding boxes and the class probabilities for each box, rather than separating the two.

For example, first the image is split into a grid and then a probability is predicted for each box within it, each with its own class probability. The class probability can then be used to predict whether or not there is an object based on a predicted value and the threshold value set. However, although this method is highly concise and efficient, it can make predicting small objects difficult as was seen in testing the ICPR 2012 mitosis dataset without proper preprocessing. On the other hand, the model can be many times faster than Faster RCNN, while still maintaining accuracy. This is likely why we saw the exceedingly high scores with each version of YOLO, even when compared to the former highest scores found with the Multi-stream Faster RCNN. That being so, YOLO was an excellent option to explore further here.

3.9.3 YOLOv5

YOLOv5 [72] is built similar to and based off of YOLOv4 except for it is built in PyTorch framework, while YOLOv4 was originally built in the Darknet Framework. Although, YOLOv5 is similar in architecture and built atop former YOLO models it included different authors and is still under debate whether or not it should be called named YOLOv5 (only to differentiate itself from YOLOv4). There are a number of different model versions and scales that have been developed specifically for YOLOv5 and the YOLO architecture itself has been continuing to improve with each iteration making YOLOv5 bar far the most advanced of the three former versions.

Some updates from YOLOv3 and YOLOv4 are listed below are only included in YOLOv5: [71]:

- There are new options for convolutional heads, including a reduced number of parameters with a faster inference speed, and an improved mAP overall.
- It uses a new approach for instance segmentation called the Path Aggregation Network or PANet [96].
- The Cross-Scaled-Partial connections have been included in the backbone help to improve speed, size, and accuracy.
- It is built in PyTorch framework, while YOLOv4 was originally built in the Darknet Framework.
- More other/newer updates including a Pytorch version of older YOLO models and more aspect scales.
- It includes Mosaic Data Augmentation.
- It includes auto learning bounding box anchors.

YOLOv5 Scaling Some models and scales were designed work better for some applications than other models. This is why there are three to four scales to set the aspect ratio of the predicted objects, which is likely what helped YOLOv5 to better localize the sizes and scales of mitosis in around half the training time of YOLOv3. It also was significantly faster to reach the optimal score on the 2014 mitosis counting dataset than YOLOv4-scaled and especially YOLOv3.

Other YOLOv5 Improvements In fact, the PyTorch framework gives the user the capability to halve the floating point precision in training and inference from 32 bit to 16 bit precision in the YOLOv5 models when used on the V100 or T4 GPUs used here. In addition, the YOLOv4 and YOLOv5 models address gradient issues such as the vanishing gradient problem, by using Cross Stage Partial (CSP) Networks for the convolutional neural network backbone, with the PA-NET neck [96]. This feature is based on the DenseNet architecture [61] which also helps the network to reuse features and helps to reduce the number of required network parameters. However, running a Tesla P100, the inference times were reduced from 50 frames per second (FPS) with YOLOv4 to 140 FPS with YOLOv5 (even after YOLOv4 had been converted to the same Ultralytics PyTorch library [72].)

YOLOv5x6 YOLOv5 version six even adds on a fourth anchor scale ratio as shown in 21. As described in the results, ICPR 2014 could not be tested with the YOLOv5x6 because the cloud runtime disconnected after 20 hours. This is likely due to the size and scale of the model; it takes much longer for the network to check an additional aspect ratio for each iteration. That being said, it is very clear this model would not be more useful in mitosis counting because the runtime is too long without any increase in accuracy. Although these version gave higher accuracy in COCO, they do not help us here because the additional scale is unnecessary. For both datasets the F-score also actually decreased when the version v5-p6 was used.

3.9.4 YOLOv4-scaled

The scaled version of YOLOv4, YOLOv4-scaled, achieves higher precision and accuracy than former YOLO models such as YOLO, YOLOv2, YOLOv3, and YOLOv4 (and even PP-YOLO), due to its' ability to scale up in size or down in size for large or small networks as shown in Table **??**. This helps it adapt to various applications such as this case with mitosis counting. Additionally, it helps it to achieve an optimal trade off between speed and accuracy, which is likely what made it more efficient than YOLOv3 for this application. Note, as shown in the tables 23 and 25 YOLOv4-scaled was set with only slightly different anchor scales and and convolutional head than YOLOv3. However, the convolutional head is very different, with YOLOv4-scaled containing more convolutional layers, such as CSP.

YOLOv4 vs. YOLOv3 In the mitosis counting datasets here, such as ICPR 2014, the scaled version of YOLOv4 achieved a higher F-score than YOLOv3 in a much shorter amount of training time. For example, in only a little more time than it took YOLOv3 to reach an F-score of 0.84, YOLOv4-scaled reached an F-score of 0.93. This proves the usefulness of this model over former models without scaling features.

YOLOv4-scaled Improvements Two of the main reasons both the YOLOv4-scaled and the YOLOv5 PyTorch training runtime is faster is first the architecture conversion to and second the addition of the Cross Scaled Partial Networks. By addition of PyTorch helped to speed up training time ten-fold when compared to the Darknet framework, which was used by YOLOv3.

Cross Scaled Partial Networks The Cross Scaling Partial Network helps to achieve up to a fifty percent reduction in the computation. This is done using a CNN that scales up the input size and number of stages for large objects in large images or scales down for small objects, by dynamically adjusting width and depth according to the inference speed requirements in realtime. This approach that modifies not only the depth, width, resolution, but also structure of the network [14].

Data Augmentation for Fine-Tuning Beginning with YOLOv4's new additional data augmentation techniques (such as mosaic) to the updated model architecture (such as the backbone and neck), it exceeded YOLOv3 in runtime and accuracy. However, another improvement that likely helped the network learn both the original and augmented images is the ability of YOLOv4-scaled to do more of the data augmentation for fine-tuning. This is unlike YOLOv4 or YOLOv3 where most of the data augmentation was done up-front.

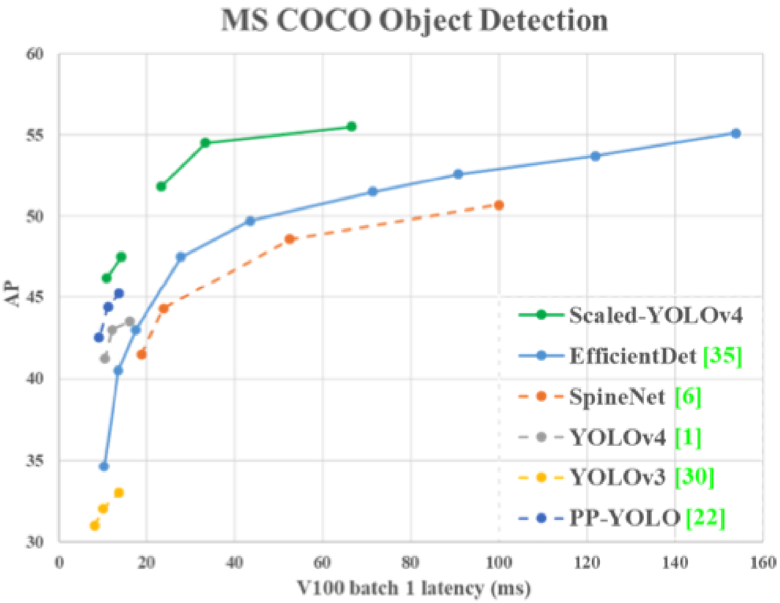


Figure 25: Comparison of the Speed/Accuracy State of the Art Models vs YOLOv4-scaled (Source[14])

On the other hand, it took almost an hour an a half to train; and, it did not exceed the YOLOv5 model in runtime for training on either dataset. Note, it was not compared to YOLOv5 in this analysis in Table 25.

3.9.5 YOLOv3

Although it took YOLOv3 over an hour and a half to reach a high F-score of 0.91, which is not quite as high as 0.94 with YOLOR, 0.93 with YOLOv4-scaled, or 0.92 with YOLOv5, it was still significantly higher than the highest F-score with Multi-stream Faster RCNN of 0.51, and it took a fraction of the training time. Clearly, each of the YOLO models are able to detect objects, such as mitosis, in real-time, much better than Multi-stream Faster RCNN. However, each year, since the development in 2016, the YOLO architecture itself has became more advanced, faster, and stronger, which is why YOLOv3 did not have the top results [128] [127] [129].

YOLOv3 vs. Faster RCNN Various incremental improvements have led to faster training times. For example, *instance segmentation* involves extracting the most detailed and finest features to identify, localize, and classify them at the pixel level and is one of the most critical tasks in image segmentation. YOLOv3 uses a Feature Pyramid Network (FPN) for the instance segmentation, which helps to detect objects of different sizes and scales. It consumes less compute power and resources than traditional pyramid networks using a multi-scale hierarchy, surpassing the original form of Faster RCNN in FPS with state-of the art accuracy and precision [91].

YOLOv3 vs. YOLOv4, YOLOv5, & YOLOR On the other hand, the neck of YOLOv4 and YOLOv5 is a PANet (Path Aggregation Network) which uses a more advanced technique called *path aggregation* and is used to help preserve more of the spatial information in instance segmentation [96]. Since the complexity of the features in a CNN increases as the image passes through the network as spatial resolution of the image decreases, the pixel-level feature masks are extracted in layers far from the the deeper layers of the network.

A FPN, on the other hand, uses a top-down path through the CNN layers to extract and combine the semantically rich features with the precise localization information. Unfortunately, this can be time-consuming for large objects or large networks because the information must be passed on through hundreds of layers. Wherefore, PANet takes a short-cut connection from both a bottom-up path as well as the top-down path originally taken by FPN, as shown in Figure 26. This makes for clean short cut paths from upper to lower layers. Usually this *shortcut connection* is only around ten layers.

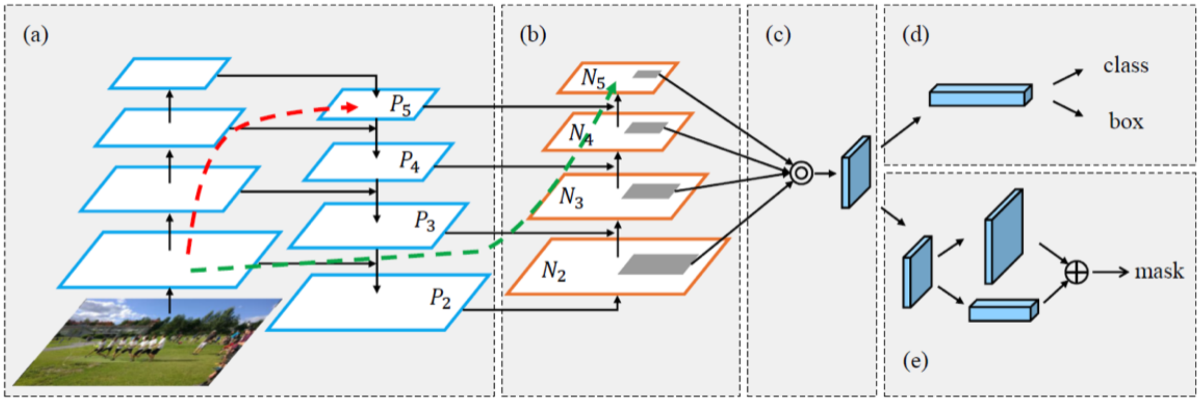
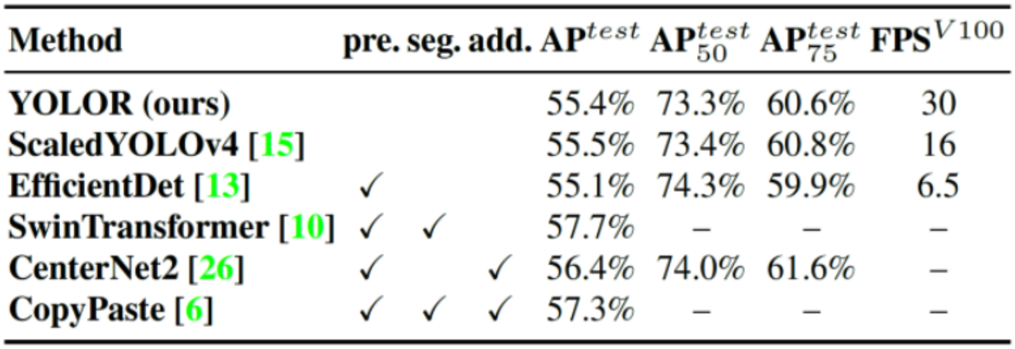


Figure 26: PANet: Path Integration Network Architecture (Source: [96])

3.9.6 YOLOR

Overall the tests here, using YOLOR [168] on any training and testing dataset was superior in terms of runtime and prediction accuracy versus any other YOLO model, besides YOLOv5. YOLOv5 came close in runtime and accuracy but it was not quite as fast. Development of the framework continues to improve with time and further iterations, with YOLOR being the most recent. YOLOR was designed to be even faster than any other former YOLO model or other object detection method, for that matter. It was shown to be able to process over four times as many FPS (Frames Per Second) as EfficientDet still with and equal or higher accuracy [168]. It also almost doubles the FPS of YOLOv4-scaled as shown in the Table 9 from [168] below.



Method	pre.	seg.	add.	AP^{test}	AP_{50}^{test}	AP_{75}^{test}	FPS^{V100}
YOLOR (ours)				55.4%	73.3%	60.6%	30
ScaledYOLOv4 [15]				55.5%	73.4%	60.8%	16
EfficientDet [13]	✓			55.1%	74.3%	59.9%	6.5
SwinTransformer [10]	✓	✓		57.7%	–	–	–
CenterNet2 [26]	✓		✓	56.4%	74.0%	61.6%	–
CopyPaste [6]	✓	✓	✓	57.3%	–	–	–

Figure 27: Comparison of State of the Speed/Accuracy of Art Models vs YOLOR(Source: [**?**])

This was somewhat expected due to the enhanced and upgraded design of YOLOR *eg. You Only Learn One Representation*, specifically as a *representation* model. With the increasing availability of data in our reach today, GPUs and machine learning networks are more likely to be developed with the ability to process much more data in a shorter amount of time. There are a number of different ways that YOLOR was developed to meet these desirable requirements.

YOLO-Representational vs. Former YOLO Versions The difference between YOLOR and previous YOLO models has to do with implicit versus explicit knowledge. Similar to a real life situation, in machine learning, implicit knowledge is learned through experience, whereas explicit knowledge is mainly learned from telling a human or machine learning network exactly what something is or direct observation. YOLOR has an unified network architecture which combines the two in order to optimize the Kernel Space Alignment, multi-task learning, and prediction refinement for learning implicit features [168]. A Feature Pyramid Network (FPN) is then used to perform multiple representations of the image so the output kernel can be aligned for simultaneously representing and detecting objects of different scales and rotations.

3.9.7 YOLOR & YOLOv5m-p5 in Parallel

?? Since both of these models achieved the top scores in the shortest runtimes, running them in parallel was superior. Both consistently produced the highest scores since each predicted slightly different. YOLOv5m-p5 model helped to increase precision in YOLOR by eliminating false-positives, while keeping the runtime low.

Table 49: Performance Comparison on ICPR 2014 Test Set

Model	F-score	Inference Time (s)
CasNN [22]	0.482	4.62
Lightweight RCNN [88]	0.427	0.83
DeepMitosis (DeepDet+Seg+Ver) [85]	0.437	0.72
FRCNN [123]	0.503	0.58
MS-FRCNN [179]	0.507	0.55
Cascaded w/ U-net [98]	0.576	-
Faster R-CNN and deep CNNs [**?**]	0.691	-
Deep Cascaded + HC [143]	0.900	0.300
YOLOv5/R	0.950	0.110
MITOS-RCNN [123]	0.955	0.500

Comparison to Other Models

Accuracy The F-score is achieved is significantly higher than contest winners score of 0.356. Additionally, Table 49 shows a comparison to some of other top-performing groups. The YOLO-based model also achieved at least as high of an F-score as others who have trained and tested their models on test sets extracted from the ICPR public training dataset. [123] and [143] both trained their model on all 3 datasets, so the resulting F-scores are not necessarily comparable.

Further, most of the top performers in literature (such as those listed below) either used a version of a regional CNN (eg. RCNN) or a deep cascaded network (as the contest winners did).

Time Analysis The RCNN-based models above require multiple hours to train [179]. For example, the Lightweight RCNN approach still requires 11.4 hours to train. Without multiple GPUs some frameworks would even be infeasible, such as, [123] which requires 5 Tesla NVIDIA K80 GPUs and 3 parameter servers. This is better than the fully CNN-based approaches initially proposed in the ICPR contests which required days to weeks to train even with a GPU [30]. However, neither type of framework is appropriate for clinical use. On the other hand, the YOLO-based real-time model only takes 15-30 minutes to train on ICPR 2012 and around and hour on ICPR 2014. Not only is the training time far shorter than other models, but the inference time per full HPF is only about 0.11 which is significantly shorter than other models as shown in Table 49.

3.9.8 Combining Datasets

The highest F-scores on the test sets were obtained by combining the two datasets, which proves the true potential for the use of deep learning frameworks for mitosis counting. If continuously adding data from other databases helps improve the prediction accuracy on any given test dataset, we would be able to keep updating the model weights by training or fine-tuning with new datasets for better results. Here, only the ICPR 2014 and combined datasets were tested. However, if the F-score continues to improve with the use of additional or alternate datasets, it is a very promising technique for use in true clinical applications. The more variety in the training examples there are, the better the network learns the features of mitosis and is able to adapt to the slightly different features contained in the test set.

3.9.9 Combining Scanner Data

Predicting One Scanner with Both Scanners By combining both the Aperio XT and the Hamamatsu Nanozoomer 2.0-HT data in the training dataset, the prediction accuracy maintained the same value in some test cases and decreased in other test cases. For example, with ICPR 2012 data, when the scanner data from both the scanners was combined in the training dataset the network predicted the Aperio XT slides alone at a slightly higher F-score, than the combined Aperio XT and Hamamatsu Nanozoomer 2.0-HT data test set. Therefore, similar to the ICPR 2014 dataset test sets, the combined training set likely predicted the Hamamatsu Nanozoomer 2.0-HT slide scanner' data at a lower accuracy than the Aperio Scanscope XT. Although the Hamamatsu Nanozoomer 2.0-HT has better resolution, the slides were all slightly darker as shown below in Figure 28. On that account, the difference in F-score may have been due to the fact that the training set had much more brighter images because they were added in the augmented dataset (eg. with the *brightness* augmentation type which was chosen to be applied over other types).

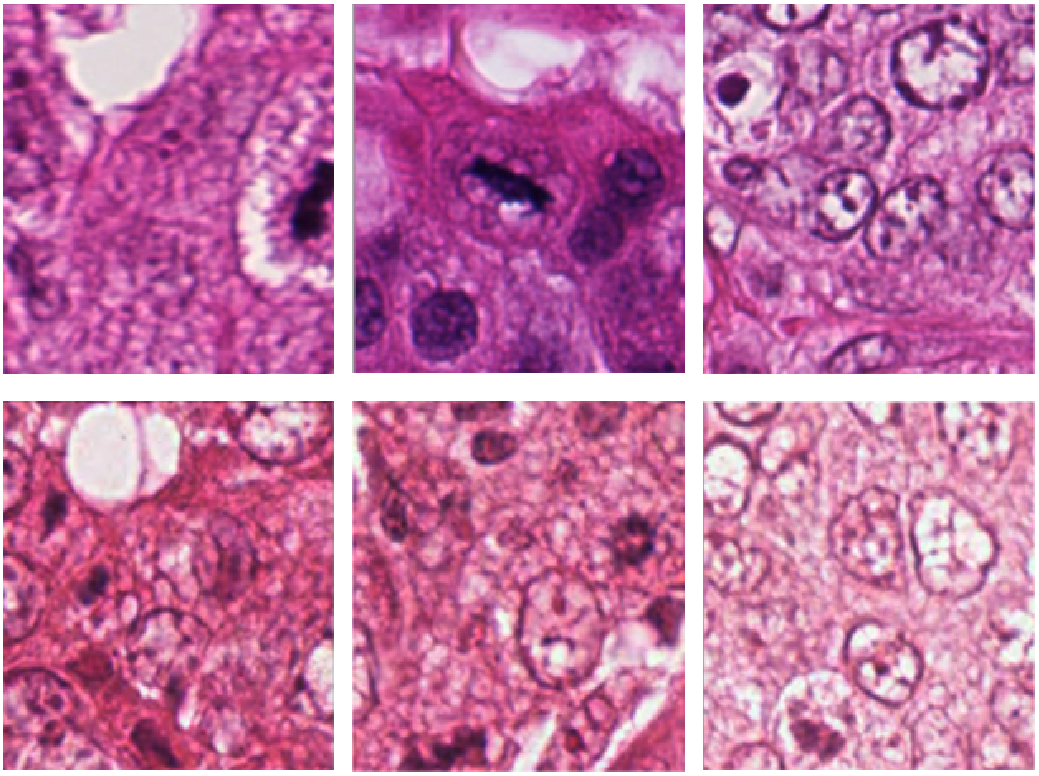


Figure 28: Row 1: Hamamatsu Nanozoomer 2.0-HT Scanner Cropped Images, Row 2: Aperio Scanscope XT Scanner Cropped Images

PREDICTING APERIO SCANSCOPE XT WITH BOTH SCANNERS VS. ONE Adding the Hamamatsu Nanozoomer 2.0-HT slide scanner data to the training set did not help in prediction in the tests on the Aperio Scanscope XT scanner data, in the ICPR 2014 dataset. The combined data training set produced the *same* F-score as with training with Aperio Scanscope XT scanner data alone. Similarly, adding the Aperio Scanscope XT scanner data slide scanner data to the training set did not help in prediction in the tests on the Hamamatsu Nanozoomer 2.0-HT scanner data; the combined data training set produced the a *lower* F-score than with training with Aperio Scanscope XT scanner data alone. This is definitely something that should be further tested with other datasets consisting of a combination of multiple scanners. If it is consistently the case, it would be much better to train with only one scanner when testing on only one scanner, because the training time drastically decreases by removing half of the training data.

PREDICTING ONE SCANNER WITH THE OTHER As far as testing the opposite dataset from which it was trained, something very interesting happened. When textitonly the Hamamatsu Nanozoomer 2.0-HT slide scanner data was in the training set and the Aperio Scanscope XT scanner data slide scanner data was tested, the F-score got to its' lowest (in these tests) of 0.81. What is especially noteworthy is that there was an opposite reaction when the opposite was tested. In the second case, only the Hamamatsu Nanozoomer 2.0-HT slide scanner data was used for training, but the Aperio Scanscope XT scanner data slide scanner data was tested. The Hamamatsu Nanozoomer 2.0-HT scanner actually helped produce slightly higher F-score than when *either* the combined scanner dataset *or* the Aperio Scanscope XT *only* training set was used.

3.9.10 Data Augmentation

Impressive Results Adding the augmented data to the training set was another way to include more training examples and feature variety for the network to learn from. This not only increased the total size of the dataset but helped to introduce more variety for the network to learn from so that if instances similar to those augmented came up in the test set, the network was better able to predict those correctly. The F-score increased very significantly by augmenting the training data in both ICPR 2014 and ICPR 2012.

For ICPR 2014, it helped to increase the F-score by 0.17 to 0.94 compared to an F-score of 0.77 with the same train and test set without the augmented data included for the same number of epochs.

For ICPR 2012, it helped to increase the F-score by 0.28 to 0.96 compared to an F-score of 0.68 with the same train and test set without the augmented data included. Likely, the ICPR 2012 dataset benefit more from the augmentation because the original ICPR 2014 dataset is two to three times larger than the original ICPR 2012 dataset .

Also, note, that when the original dataset (without augmentation) training time was increased to the same training time as adding the augmented data (eg. by doubling the number of epochs) the F-score only increased by about 0.1 (for each dataset). Therefore, the data augmentation up-front was still crucial to obtaining the very high F-score.

Comparison of Differences in Augmentation Types Interestingly, for the ICPR 2014 dataset, with the addition of any of the tested types of augmentation the model produced a very similar F-score on the test set, with each only differing by about 0.01. On the other hand, with the ICPR 2012 dataset, exposure worked slightly better than the others and mosaic slightly worse. The larger difference is also likely due to the higher significance of the augmented data with ICPR 2012 and to the fact that the YOLO framework is built with data augmentation techniques within. For example, YOLOv4 and on introduced new data augmentation techniques such as mosaic and cut out. Possibly, the combination of two mosaic augmentation application compounded upon one another reduced the networks ability to learn from those examples because the region of interest became to small for the network parameters. In this case, the original input images already had mosaic applied so when they were input into YOLOR they were even more *mosaicly* augmented. For example, four images would have became eight different images, so the part of the bounding box would be cut off from most of the images.

The exposure (and brightness to an extent) data augmentation types likely had a slightly *better* result because HPF scanner slides often have these types of changes in real-life. This is likely an actual occurrence that happened in the test set, and the network was better prepared. Parameters that can contribute to variations effecting representation of the HPF include scanner optics, camera sensors, and digital resolution, scan resolution, image viewer, monitor size, aspect ratio, and display resolution. [76].

AUGMENTED DATA WITH MULTI-STREAM FASTER RCNN When the augmented dataset is included, the size of the training set increased two to three fold, making the training runtime nearly two to three times longer for a given number of epochs and set of parameters. This would have been an issue with any other training network besides YOLO (*especially* YOLOR) which is designed to be fast, dynamically modifying the size and scale for an optimal tradeoff between speed and accuracy. As described above, the YOLO *You Only Look Once* model is a realtime one stage object detector, which looks all at once and makes the predictions globally among the full image context. This is unlike Faster RCNN, a two stage detector, requiring multiple networks to generate predictions and most often making it tens to hundreds of times slower. Neither the Faster RCNN model or the Multi-stream Faster RCNN model would have been able to complete the training within the allotted GPU time since it already takes hours to train the un-augmented dataset alone on either. That being said, based on the effect data augmentation had when using YOLO, not only in these tests but within the YOLO framework itself, augmenting the data prior to the Multi-stream Faster RCNN likely would have made a dramatic difference in the results, as well.

YOLOV4 AND UPWARD INTERNAL DATA AUGMENTATION One of the key improvements to the YOLOv4 (and upward) models is the contribution of a *"Bag of Freebies"*, which is a nickname for the set of techniques added to the framework scale up the dataset with augmented images. Although, they *do* add to training time required they *do not* add to inference time. They also helped YOLOv4 (and on) increase in accuracy (AP) 10% on the standard COCO dataset. Therefore, the implementations of YOLOv4, YOLOv5, and YOLOR (used in these tests) already include augmentation techniques similar to those also known as Photometric Distortion, Geometric Distortion, Random Erase, Hide and Seek, and MixUp. These are similar to the techniques employed in the tests here in the results section. For example, *Photometric Distortion* includes changing the brightness, contrast, saturation, and noise in an image, and four of the augmentation techniques tested were blur, noise, exposure and brightness. *Geometric Distortion* is another example, which includes random scaling, cropping, flipping, and rotating. Rotation is another technique that was tested. *Mosaic Data Augmentation* was also tested here and is directly employed in YOLO already.

The fact that so many augmentation types are already included in the framework itself, is likely the reason why doubling the number of training epochs slightly improved the accuracy from the un-augmented dataset (but not the *augmented* dataset). It is also likely the reason why adding image data to the training set with any type of augmentation resulted in a similar increase in F-score, when compared to another type of augmentation, in ICPR 2014. However, note, that YOLOv4 also includes a variety other data augmentation techniques such as Self-Adversarial Training (SAT) for the detector, as well as class label smoothing, DropBlock regularization, and Cut-Mix in the backbone, which were not included in the additional steps added up-front [14].

3.9.11 Changing the Data Preprocessing Method for ICPR 2012

Preprocessing ICPR 2014 As stated in 3.7.1, the ICPR 2014 dataset was trained with the cropping of the images in the training set from the original HPF into 64 (8×8) equally sized subsections. For example, since the original ICPR 2014 Aperio Scanscope XT scanner image is 1539×1376 pixels the original cropped images are of size 192 by 172 pixels. Each HPF 1539×1376 pixel image produces 64 images. This is done so that the least amount of data is left unused in training. Similar processing can be done with the Aperio XT Scanner images from the ICPR 2012 dataset except they are size 2084×2084, resulting in 64 cropped images sized 260×260.

Scaling Requirements for ICPR 2012 However, the cropped images for ICPR 2012 produced had an area of localization too small for detection by the network. Since the original image size is slightly larger than the ICPR 2014 dataset, the cropped locations were slightly larger and therefore the mitosis and bounding box coordinates were also slightly smaller then ICPR 2014. Even after the images are resized to 416×416 the bounding box is too small for detection because the bounding box coordinates adjust in size at the same rate as the actual image (without further cropping.)

Adding a Sliding Window This was further verified after adding an increment in the sliding window when cropping the original images from the HPF slides. Even after doubling or tripling the size of the training set (before augmentation) by cropping out more images from different angles these were still inadequate for proper training. The resulting in a maximum F-score of 0.88 because some of the mitotic regions became to small for the network to find. Interestingly, adding the sliding window increment did not have an effect on the F-score.

Finalized, Compromised Solution As described in 3.7.1, for this reason, the images from the 2012 dataset were cropped smaller than the ICPR 2014 images. The ICPR 2012 dataset was trained on a dataset created with the cropping of the images from the original HPF into 128 (16×16) equally sized subsections to preserve the maximum amount of the image and create sub-images of similar size.

Focus/Resolution Differences Between Datasets in Combined Data Tests While this helped to provide more data for training the combined datasets, it may have offset the network since the bounding box coordinates for mitosis and the mitosis itself were still slightly different size. Furthermore, the training images for the YOLO setup are required to all be the same size when input into the network. Therefore, the ICPR 2012 images were marginally different in focus or resolution from the ICPR 2014 dataset images. And, the closer in size each mitosis, the better the network can learn from the training examples.

3.10 CONCLUSIONS

It is crucial for clinicians to provide important prognostic information at the time of a cancer diagnosis for proper treatment and monitoring [148]. The machine can process a vast amount more data than a human can and it can do this task in a much shorter amount of time. Additionally, state-of-the-art object detectors can easily make course pixel-level predictions with precise-level scores [65], whereas some of the differences in mitotic figures can be indistinguishable to the human eye.

In this section, a couple of new models for mitosis counting were proposed, achieving state-of-the-art results. First, a segmented image map generated with UNet and is the input to a second stream of the Multi-stream Faster RCNN so that features from both streams are fused in the bilinear pooling layer. The additional detail features from segmentation help to localize the mitotic regions and improve the accuracy over other methods and over a single stream alone. An F-score of 0.508 was obtained, which is higher than both the winners score and scores from other methods on the same data. Additionally, the technique is clinically applicable taking less than a second to test a single HPF. However, the speed and accuracy of newer state-of-the-art models such as YOLO are now leaders in object detection, which had yet be applied to mitosis counting.

Second, YOLO-based [128] models were tested as a tool for mitosis counting. Multiple models and versions of YOLO were compared with different types of augmentation, and then top models were run in parallel for superior results. The model out-performed all earlier methods on the ICPR contest datasets when YOLOR [168] was run in parallel with YOLOv5m-p5 on two separate cloud GPUs with exposure augmentations added and the results were averaged. Using these techniques the highest F-scores of 0.95 and 0.96 on the MITOS-ATYPIA 2014 challenge and MITOS-ATYPIA 2012 challenge mitosis counting datasets are achieved, respectively. Additionally, it took a fraction of the time for both training and testing, making it clinically applicable.

With the help of automatic image detection systems, like the YOLO-based model, the possibilities for better medical care in the future are endless. This is especially the case for cancer detection and/or assessment, since many types of cancer are analyzed using imaging techniques, in particular histopathology images, as are used here in this study. Even if the deep-learning-based detection tool is not the final in-clinic prediction, it could be extremely useful as a second opinion or to mitigate risk until it is further tested or more trusted for use with deeper integration in human lives [175].

3.10.1 FUTURE WORK

IMPROVING MULTI-STREAM FASTER RCNN Based on the increase in the F-score and the amount of decrease in training time of the YOLO models, it is not recommended to continue work on the Multi-stream Faster RCNN model. There are newer modifications to this network itself, as well as newer frameworks for deep learning which have far exceeded the Faster RCNN framework architecture. There are many different upgrades and changes that would be needed to be made to the network prior to its use. Consequently, it wouldn't be worthwhile to spend the time to prepare all of those changes. There are too many to make before it comes anywhere close to new models in its runtime and accuracy abilities. Below lists some examples of those.

DATA AUGMENTATIONS The Multi-stream Faster RCNN experiments did not include any type of augmented data in the training set. Augmentation techniques are especially useful and can significantly improve accuracy when the size of the training data is small [84] [86] such as in the case with mitosis counting. Data augmentation techniques such as rotating, mirroring, resizing, brightness, contrast, saturation, mosaic, and noise or blur are all examples of techniques that could have been applied during data preprocessing to increase the size of the training set. Their use is very likely to improve upon the highest possible F-score that can be obtained with this model.

For example, these augmentations dramatically improved the results in YOLOv4 [14] versus YOLOv3 and lower models, as described in 3.9.10. For another example, the Python **Albumentations** [18] library was used with EfficientDetdx in top scoring implementations (eg. [40]) in the Kaggle Global Wheathead Detection contest [6] which is a similar type of challenging object detection task. **Pseudo-labeling** or cropping out images containing mitosis cross boundaries can also help provide more training samples to the network. However, as noted in 3.9.10 due to the structure of Faster RCNN, it would also require a far longer runtime and/or a much more powerful GPU to process all of this addtional data; it is unquestionable that any version of YOLO would still be a faster an more precise object detection model, in any case.

WEIGHTED BOXES FUSION In the Wheathead Contest Dataset, a Faster RCNN model has been tested with **WBF** (Weighted Boxes Fusion) [149] in place of NMS or SoftNMS (based on [147]). WBF is a method used to combine the boxes proposed in object detection models in order to make a better prediction overall, rather than removing some of the proposals from one model alone. The confidence scores of all proposed bounding boxes to combined to find the averaged boxes for prediction. This is another integration that is likely to enhance performance in mitosis counting by merging predictions from separate models.

ADDING MORE STREAMS Due to the fact that the addition of a second stream (the segmentation stream) helped to improve results for image tampering and mitosis counting, it is likely that the addition more streams will improve the F-score further. For example, the combinations of features from other classic mitosis segmentation models will likely produce further improvement.

MASK RCNN Mask R-CNN [59] is an extension to Faster R-CNN which adds a parallel branch for instance segmentation with the object detection branch. This branch adds a branch that outputs a binary mask that says whether a pixel is part of a given object for more precise prediction. This helps with the pixel to pixel alignment requirement which is missing from the normal Faster RCNN for a more clear-cut localization of objects. For example, Faster RCNNs' RoI Pooling layer performs coarse spatial quantization for feature extraction which fails to preserve the exact spatial locations of objects and could lead to inaccuracies. Another benefit is Mask RCNNs' ability to decouple the mask and class predictions to better predict the category, rather than create a class competition. On the other hand, the fully connected network in Faster RCNN couples predictions. This Mask RCNN extension could be added to the Multi-stream Faster RCNN as well, in order to improve prediction accuracy with little addition to overhead computation.

FEATURE PYRAMID NETWORK In Faster R-CNN, a Region Proposal Network (RPN) with a single-scale feature map is used for region proposals. As previously described, the network consists of a single 3×3 convolutional layer followed by two sibling 1×1 layers. A small subnetwork performs the binary classification and bounding box regression and it is evaluated on dense 3×3 sliding windows, on top of a single-scale convolutional feature map. However, detecting objects in different scales is more challenging for small objects.

Processing multiple scale images is time and resource consuming because a pyramid of images is required for each feature extracted. However, an FPN (Feature Pyramid Network) [91] helps greatly improve the accuracy and speed. In place of the feature extractor, it generates multi-scale feature maps from multiple layers. It uses a bottom-up and top-down approach on the feature pyramid to create a pyramid of feature maps. These contain with higher quality information and can be later fed into the detector. Note, FPN can also help to extract masks for image segmentation, similar to Mask RCNN.

NETWORK LAYERS & BACKBONE Modifications to the network layers and/or parameters will also likely lead to additional improvements [143] [85]. The convolutional neural networks (CNNs) are the backbones of the object-detection model are the key to extracting features and their design has been heavily researched. Some of the most popular CNNs include AlexNet, VGGNet, and ResNet, and the number that usually precedes is the number of layers. However, not all of these are equal. The original Faster RCNN used a VGG-16 backbone rather than ResNet (the Residual Network), *which was used in the Multi-stream version here.* When used in Faster RCNN, the Inception-ResNet models have led to better accuracy performance at shorter epochs [11]. VGG-16 [97] is composed of 13 convolutional and 3 fully connected layers with ReLU activation. Whereas, Inception-ResNet-V2 [151] is composed of 164 deep layers and about 55 million parameters.

TRAINING & TESTING ON OTHER DATASETS The authors who have produced some of the highest scoring implementations for the mitosis counting datasets reached this top score by combining the training sets for all three of the most popular mitosis counting contests; the MITOS-ATYPIA 2012 challenge [137], the MITOS-ATYPIA 2014 challenge [136], and the AMIDA 2013 challenge [163] [123]. Next, it will be interesting to test the combination of even more datasets and to compare the score. It may likely be the case that similar to testing the combined MITOS-ATYPIA 2012 challenge dataset and MITOS-ATYPIA 2014 challenge dataset (as was done here) that the combination of all three datasets produces the best results on the test set thus far for any challenges' test set.

Additionally, the other two mitosis counting contest datasets should be tested alone, as well as in combination with the other. Pre-training with two of the other datasets and then fine-tune with only one dataset is another potentially useful way to carry out the task that could be applied. This technique helped to greatly improve accuracy in the noise version of Bi-linear Faster RCNN, so it may be helpful here as well, both with YOLOR and Multi-stream Faster RCNN [187].

CONSISTENCY IN SCORING WITH MULTIPLE DATASETS As further discussed in 3.10.1, precautions must be taken into account when using different microscopes or scanners. For example, proliferative activity is used to score by counting mitotic figures within hotspots, which are areas with the highest number of mitoses within a tumor. Ten HPF are typically counted in a hotspot region with a light microscope. However, different microscopes have different field areas, which vary between 1.26 to 3.74 mm^2 per the 10 HPFs. Therefore, in order to be consistent among microscopes a detailed table must be used to obtain the mitotic score for the field area [63]. Data augmentation, as was used here, will also likely make the model more robust toward these differences.

TUMOR PROLIFERATION ASSESSMENT CHALLENGE 2016 (TUPAC16) Faster methods of of whole slide imaging technology in digital pathology is becoming an increasingly popular for routine tests, but large-scale implementation can be technically and financially challenging [66]. A similar research challenge dataset from the TUmor Proliferation Assessment Challenge 2016 (TUPAC16) assesses tumor proliferation from Whole Slide Images (WSI) [**?**]. There are 500 training images and 321 testing images and the goal is to determine the proliferation rate based the mitotic score (similar to the other contests). It would be an interesting challenge to see whether the networks developed and tested here in this work could be applied directly or modified to be adapted to this type of task. Since the region of interest is different (since the WSI is much larger) there would either need to be new training done to detect many mitosis in a region or the image would have to be cropped significantly (which may result in a loss of the image information).

WSI MODIFICATIONS & COMPARISON Consequently, this task may involve selecting tumor areas with the greatest number of mitosis, known as *mitotic hotspots*, which is another time-consuming and error-prone task when done by human pathologists. Based on numerous studies in the reproducibility of mitosis counting this task is also highly applicable to automation [117]. For example, Tabata et al. evaluated the ability for pathologists to accurately make predictions of mitotic figures from WSI in comparison microscopes in order to determine opportunity for algorithms to make automatic quantifications. They found that mitosis detection accuracy for 75% WSI scanners was significantly less than that of the microscope (with accuracy was between 0.632 and 0.843) [152].

However, some groups argue that mitosis counting should be done specifically with WSI [161] rather than other methods. Therefore, it would also be interesting to compare the usefulness of the deep learning methods for this type of task of counting mitosis from WSI versus the task of counting mitosis from High Power Fields (HPF). Another question to explore is whether or not the data from both types of tasks could potentially be somehow combined to produce superior results in mitosis counting.

THE NEED FOR MORE PUBLIC MITOSIS COUNTING DATASETS Tests on more datasets for mitosis counting or for other applications could help to prove the full true potential for this model to be used in a clinical setting at some point. However, one major requirement for future work is use of a GPU which can handle more data and the longer runtimes required for these types of tests. It would be most interesting to include more and more datasets to see how well the model can predict various test sets using YOLOR. The more variety in the test sets and training sets, the more likely the model is to be able to generalize to new samples in a real life or clinical setting for example.

YOLO FUTURE WORK Overall, the YOLOR model was the YOLO framework version (and the overall best deep learning framework) with the lowest runtimes and highest F-scores due to its high degree of advanced architecture. This framework has been built upon the greatest techniques of the most recent work done in the field and has been through the most iterations of development. On account of the impressive results obtained here, it is strongly recommended this representational YOLO model be used for any new similar research tests in this area. This further testing should at least include the exploration of those questions listed below:

- Combining scanner data: *Is combining the data from other scanners and/or microscopes really helpful to achieve optimal results on the same scanner or for different scanners used in the creation of the test set (as was the case here)?*

- Combining datasets: *Can combining multiple datasets really produce higher scoring results on new test data (as was the case here)?*

- Other medical applications: *Can this technique be modified slightly to be trained for other (similar) medical applications?*

COMBINING SCANNER DATA Combining the data from multiple slide scanners can be a better way to increase the number of images in the training dataset without augmentation. If multiple scanners are available, there is more variability in the colors and pixel textures from each slide for the network to learn from. Tissue slides may vary greatly in appearance due to their acquisition from different pathology labs and preparation protocols. Consequently, adding data from other scanners could help to develop a generalized model for use on test sets created with other scanners (besides those already in the training set).

SCANNER DIFFERENCES For example, simply based on the microscope that is being used at the time a pathologist may call a certain tumor malignant, the other may call it benign [31]. High power fields produced by microscopes may differ significantly, even at the same high power $(\times 400)$ magnification [31]. Therefore, automatic image processing should also take into account the differences in the microscopes used and the slide scanners. However, it is highly important to make proper use of the HPF size in order to pre-process data equivalently among datasets and for proper bounding box size annotations. For example, as was seen in the results, tests with datasets constructed with bounding boxes of improper size according to the network scales will lead to missed predictions. Additionally, when multiple scanner data subsets are combined it is crucial for the dimensions and magnifications of mitotic figures to be equal among each subset so that the network does not need to spend time learning mitotic figures of multiple different scales. Using the incorrect size and scale for annotating mitosis could also lead to misdiagnosis [31]. Additional data augmentation techniques will likely help increase the variability in the training set to make it more robust toward new test sets with alternate scanners.

[63]

COMBINING OTHER DATASETS Combining datasets is another way to introduce variability in the dataset in order to generalize to new test set data (eg. from other datasets besides ICPR 2012 or 2014). As described above, top prediction scores were obtained when the two datasets were combined in the training set. Thus, based on the impressive results, this is a top area recommended for future work with YOLOR.

OTHER MEDICAL APPLICATIONS It is highly recommended that YOLOR (or YOLOv5) be undertaken in new research for not only for use in supplemental mitosis counting studies, but for *other similar medical applications*. There are many uses of object detection in the medical field, especially with the vast amount of image data being generated today. Now that real-time automatic detection is available, YOLOR should be explored for more applications where a quick prediction (or initial estimation prior to a pathologists') is required. Imaging technologies such as MRI and CT imaging capture 3D spaces contain inherent spatial separation so there is somewhat less of need for discrimination between overlapping objects.

On the other hand, clinical applications such as radiation therapy planning or tumor growth require monitoring pixel-wise predictions in order to localize objects of interest [65]. For example, brain tumor detection and segmentation through the Magnetic Resonance Images (MRI) is a challenging task as the brain tumor varies in size, shape, and structure [122]. Due to the short supply of doctors' expertise, an automated system is needed to extract the specific features. Also, Wulczyn et al. developed a deep learning system to predict disease specific survival across ten cancer types from The Cancer Genome Atlas (TCGA) based on histopathology images [175]. Their approach demonstrated the potential for this approach to provide exceptional prognostic information in numerous types of cancer, and even within different pathological stages. This group also highlighted that future work will benefit from larger data-sets gathered for the purposes for improved survival modeling.

However, there has even been some research done with the use of former versions of YOLO (eg. YOLO, YOLOv2, and YOLOv3) showing great potential in being applied to localization in CT scans and other types of imaging, as well. For example, radiographic evidence of Pneumonia was detected using YOLOv3 from chest scans [180], detection and localization of lung cancer was implemented using YOLO to detect the nodules in CT scans [139], and YOLOv3 on kidney localization in 2D and in 3D from CT scans [**?**]. The YOLO framework could also be applied to detection and segmentation of either artifacts or diseases in endoscopic images [112] and detection of diabetic foot ulcers [53], as other deep learning networks for localization have.

4 Overall Conclusion

In the age of Big Data, the amount of digital information we possess on our limited size machines continues to increase. Further, the potential to improve machine learning continues to increase simultaneously and at an even faster rate as we gain more data and build better machine learning algorithms and deeper networks. These highly-automated alternative methods continue to prove more and more useful over their classic, hand-carved, or manually designed feature extraction method versions. This is especially the case when the features are combined. As shown here, the accuracy of some of the machine learning or deep learning tools can be improved by combining them with features that have been extracted from another method. Additionally, the computational time can then be decreased by running them in parallel or running the same network on multiple nodes. In this work, these techniques were carried out in multiple domains to produce faster and more accurate tools or frameworks to state-of-the-art results. The summary of the overall contributions for each of the three main sections of this dissertation are listed below.

4.1 Summary of Contributions

Parallelizing Machine Learning Algorithms

- A method called Software Alchemy (SA) was introduced and adapted to many different machine learning algorithms. In cases where it was not straight-forward (eg. NMF) a technique or tool was provided to facilitate the use of this method.

- Substantial experimental results/analysis were produced on the overall effect on runtimes and accuracy via using more or less nodes in a cluster or data points per node.
- Software programs were written to facilitate SA or parallelizing (eg. Hyperquicksort, Partools functions for node-to-node communication). See `Partools Github`.
- SA was applied in other new ways such as to multiple GPU's, deep learning, and to CUDA GPU streams.
- SA was adapted to image fraud detection - to parallelize some of the new algorithms described below (parts of the sliding window, RLRN, GLM, and full image processing on multiple GPUs). See `imagefraud Github`.

IMPROVING IMAGE FRAUD DETECTION ALGORITHMS Software programs for image fraud detection were written that out-performed the state-of-the-art. See `imagefraud Github`. These are summarized below:

- A new method based on Run Length Run Number (RLRN) encoding was paralellized (RLRNpar). It was modified to train a GLM model based on this feature array from a user input of an entire database of images. Reduction in runtime up t 2 hours and accuracy 20 %
- BAGlocalization uses the Block Artifact Grid (BAG) to measure differences in the compression rate and quality factor of image blocks used in JPEG compression.
- ELA runs an "Error Level Analysis" on JPEG (or JPEG converted) images to determine the difference in compression levels throughout the image and localize forgery.
- A new method called combinedLocalization combines BAG extraction and ELA to outperform the classic methods and better localize the artifact and to outperform the classic state-of-the-art algorithms.
- A new method called pcaCProbust combines an intensity based and PCA-based algorithm for image fraud, to outperform the classic state-of-the-art algorithms.
- A new method called RLRNlocalization is an algorithm for localizing tampered regions based on RLRN (which normally does not localize objects).
- A new method called dctCPtested is based on the DCT quantization algorithm for copy-paste image fraud. It uses the chrominance and luminence maps to extract copied regions, outperforming the state-of-the-art algorithm.

- The Multi-stream Faster-RCNN is a new model that adapts an object detection network to localize any type of image tampering. It improves upon the accuracy and runtimes for use of the single stream Faster-RCNN alone as well as all other image-tampering algorithms.

DEEP LEARNING FOR MITOSIS COUNTING

- A new model, the Multi-stream Faster RCNN, was built for mitosis counting, outperforming the state-of-the-art (at that time) and any other RCNN-based models. This fuses features from UNet segmentation with the RGB features to produce a more precise feature map.
- A number of different models that have never been applied to mitosis counting were adapted to the task (eg. YOLOv3, YOLOv4-scaled, YOLOv5, and YOLOR). Further, this is the first ever *real-time* model designed for this purpose. Run-times and accuracy exceeded the state-of-the art.
- Multiple versions of the model and versions were tested extensively with different types of augmentation, datasets, and scanner-versions, building up research results.
- An improved YOLO-based model was developed which combines/averages the predictions from the two top-performing YOLO versions by running them in parallel, on two separate cloud-based GPU's. Run-times and accuracy *far* exceeded the state-of-the art, since both YOLO versions produce slightly different predictions.

4.2 FUTURE WORK

4.2.1 SOFTWARE ALCHEMY

Software Alchemy, the technique introduced and used throughout Section 1, could likely be used to further improve the results in the deep learning networks used for both tampering and mitosis detection in Sections 3 and 2.

For example, either the Multi-stream Faster RCNN or the YOLO-based model could be easily be run on multiple GPUs (eg. cloud GPUs), each with smaller/sub-chunks of the data. Since the dataset is smaller, the training time would be significantly shorter, likely without loss in accuracy, as shown in 1.6.11.

There would be a decrease in training time required for a given number of epochs. This would be especially useful for slightly larger datasets such as ICPR 2014 or the combined ICPR 2012 and ICPR 2014 dataset with data augmentation, which still can take over an hour to train with the YOLO-based model. In particular, it would be extremely helpful for training the Multi-stream Faster-RCNN model which can take 6-12 hours to train on a GPU. Further, as seen in **??**, it is likely that there would not be a loss in accuracy. Since the results can be averaged by averaging the bounding box coordinates, any loss due to insufficient data on a single GPU is overcome.

Similarly, another thing that could be done (to add more improvement in accuracy rather than decrease in time) is to add more augmented data before distributing it among multiple GPUs. Even though the training time may not decrease at all or as much, there would be more training done overall. For example, in **??** the same amount of data was added to each GPU, but the accuracy was improved because the models were slightly different, producing slightly different results on each node. The average of the predictions from each trained model was superior to that of a single model.

References

[1] TensorFlow for R from RStudio imdbcnn.

[2] H. F. Elyamany A. L'Heureux, K. Grolinger and M. A. M. Capretz. Machine Learning With Big Data: Challenges and Approaches. *IEEE Access*, pages 7776–7797, 2017.

[3] Ritu Agarwal, Deepak Khudaniya, Abhinav Gupta, and Khyati Grover. Image forgery detection and deep learning techniques: A review. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1096–1100, 2020.

[4] C.C. Aggarwal. *Recommender Systems: The Textbook*. Springer International Publishing, 2016.

[5] Andrew Andrade. Map reduce with examples.

[6] Ben Hamner Anthony Goldbloom. Kaggle: Global Wheat Detection. `https://www.kaggle.com/c/global-wheat-detection/`, 2020.

[7] Ignacio Arganda-Carreras, Srinivas C. Turaga, Daniel R. Berger, Dan Cireşan, Alessandro Giusti, Luca M. Gambardella, Jürgen Schmidhuber, Dmitry Laptev, Sarvesh Dwivedi, Joachim M. Buhmann, Ting Liu, Mojtaba Seyedhosseini, Tolga Tasdizen, Lee Kamentsky, Radim Burget, Vaclav Uher, Xiao Tan, Changming Sun, Tuan D. Pham, Erhan Bas, Mustafa G. Uzunbas, Albert Cardona, Johannes Schindelin, and H. Sebastian Seung. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, 9:142, 2015.

[8] Akinfenwa. Atanda, Mohammed. Imam, Ali. Umar, Ibrahim. Yusuf, and Shamsu. Bello. Audit of nottingham system grades assigned to breast cancer cases in a Teaching Hospital. *Annals of Tropical Pathology*, 8(2):104–107, 2017.

[9] Maschenka C. A. Balkenhol, Peter Bult, David Tellez, Willem Vreuls, Pieter C. Clahsen, Francesco Ciompi, and Jeroen A. W. M. van der Laak. Deep learning and manual assessment show that the absolute mitotic count

does not contain prognostic information in triple negative breast cancer. *Cellular Oncology*, 42(4):555–569, Aug 2019.

[10] Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015.

[11] Ayoub Benali Amjoud and Mustapha Amrouch. *Convolutional Neural Networks Backbones for Object Detection*, pages 282–289. 07 2020.

[12] C. N. Bharti and P. Tandel. A survey of image forgery detection techniques. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 877–881, March 2016.

[13] Xu Bo, Wang Junwen, Liu Guangjie, and Dai Yuewei. Image copy-move forgery detection based on surf. In *2010 International Conference on Multimedia Information Networking and Security*, pages 889–892, 2010.

[14] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[15] L. Bondi, S. Lameri, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro. Tampering detection and localization through clustering of camera-based cnn features. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1855–1864, July 2017.

[16] MOMCILO BRAJIC and John Naisbitt. Overlapping block based algorithm for copy-move forgery detection in digital images. In *Semantic Scholar*, 2016.

[17] P. Bühlmann, P. Drineas, M. Kane, and M. van der Laan. *Handbook of Big Data*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, 2016.

[18] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[19] Emmanuel Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Commun. ACM*, 55(6):111–119, June 2012.

[20] H. Cao and A. C. Kot. Accurate detection of demosaicing regularity for digital image forensics. *IEEE Transactions on Information Forensics and Security*, 4(4):899–910, Dec 2009.

[21] Hao Chen, Qi Dou, Xi Wang, Jing Qin, and Pheng Heng. Mitosis detection in breast cancer histology images via deep cascaded networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Feb. 2016.

[22] Hao Chen, Qi Dou, Xi Wang, Jing Qin, and Pheng-Ann Heng. Mitosis detection in breast cancer histology images via deep cascaded networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1160–1166. AAAI Press, 2016.

[23] J. Chen, X. Kang, Y. Liu, and Z. J. Wang. Median filtering forensics based on convolutional neural networks. *IEEE Signal Processing Letters*, 22(11):1849–1853, Nov 2015.

[24] Xi Cheng, Bohdan Khomtchouk, Norm Matloff, and Pete Mohanty. Polynomial regression as an alternative to neural nets, 2018.

[25] Wei-Sheng Chin, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. A fast parallel stochastic gradient method for matrix factorization in shared memory systems. *ACM Trans. Intell. Syst. Technol.*, 6(1):2:1–2:24, March 2015.

[26] H. Choi, H. Jang, D. Kim, J. Son, S. Mun, S. Choi, and H. Lee. Detecting composite image manipulation based on deep neural networks. In *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 1–5, May 2017.

[27] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou. An evaluation of popular copy-move forgery detection approaches. *IEEE Transactions on Information Forensics and Security*, 7(6):1841–1854, Dec 2012.

[28] Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y. Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 281–288, Cambridge, MA, USA, 2006. MIT Press.

[29] Dan C. Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. *Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention*, 16 Pt 2:411–8, 2013.

[30] Dan Cireşan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. volume 16, pages 411–8, 09 2013.

[31] Ian Cree, Puay Tan, William Travis, Pieter Wesseling, Yukako Yagi, Valerie White, Menaka Lokuhetty, and Richard Scolyer. Counting mitoses: Si(ze) matters! *Modern Pathology*, 34, 06 2021.

[32] Douglas W. Cromey. Avoiding twisted pixels: Ethical guidelines for the appropriate use and manipulation of scientific digital images. *Science and engineering ethics*, 16 4:639–67, 2010.

[33] A. DasGupta. *Asymptotic Theory of Statistics and Probability*. Springer Texts in Statistics. Springer New York, 2008.

[34] Reza Davarzani, Khashayar Yaghmaie, Saeed Mozaffari, and Meysam Tapak. Copy-move forgery detection using multiresolution local binary patterns. *Forensic Science International*, 231(1):61–72, 2013.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[36] Leonid Djinevski, Sasko Ristov, and Marjan Gusev. Superlinear speedup for matrix multiplication in gpu devices. In Smile Markovski and Marjan Gusev, editors, *ICT Innovations 2012*, pages 285–294, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[37] Samuel F. Dodge and Lina J. Karam. Understanding how image quality affects deep neural networks. *CoRR*, abs/1604.04004, 2016.

[38] J. Dong, W. Wang, and T. Tan. Casia image tampering detection evaluation database. In *2013 IEEE China Summit and International Conference on Signal and Information Processing*, pages 422–426, July 2013.

[39] J. Dong, W. Wang, and T. Tan. Casia image tampering detection evaluation database. In *2013 IEEE China Summit and International Conference on Signal and Information Processing*, pages 422–426, July 2013.

[40] Liao Etienne David, Ian Stavness. EffDet-D6-PL[s—bn]-R[bb—a3—usa]-Eval[94]-13-DB. `https://www.kaggle.com/alexanderliao/effdet-d6-pl-s-bn-r-bb-a3-usa-eval-94-13-db`, 2020.

[41] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[42] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[43] Y. Feng. *Sure Independence Screening*, 2016.

[44] Thomas S. Ferguson. *Course in Large Sample Theory (Chapman & Hall/CRC Texts in Statistical Science)*. Chapman and Hall/CRC, 1996.

[45] Jessica Fridrich, David Soukal, and Jan Lukás. Detection of copy-move forgery in digital images. *Int. J. Comput. Sci. Issues*, 3:55–61, 01 2003.

[46] Stefan Fritsch and Frauke Guenther. *neuralnet: Training of Neural Networks*, 2016. R package version 1.33.

[47] Yusuke Fujii, Takuya Azumi, Nobuhiko Nishio, Shinpei Kato, and Masato Edahiro. Data transfer matters for gpu computing. pages 275–282, 12 2013.

[48] Katelyn Gao and Art Owen. Efficient moment calculations for variance components in large unbalanced crossed random effects models. *Electron. J. Statist.*, 11(1):1235–1296, 2017.

[49] Juan Garcia. Linear regression in rhadoop via mapreduce.

[50] Vincent Garcia, Eric Debreuve, and Michel Barlaud. knn cuda- fast k nearest neighbor search using gp. http://vincentfpgarcia.github.io/kNN-CUDA/, 2018.

[51] Thomas Gloe and Rainer Böhme. The'dresden image database'for benchmarking digital image forensics. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1584–1590, 2010.

[52] Manu Goyal and Saeed Hassanpour. A refined deep learning architecture for diabetic foot ulcers detection, 2020.

[53] Manu Goyal and Saeed Hassanpour. A refined deep learning architecture for diabetic foot ulcers detection. *CoRR*, abs/2007.07922, 2020.

[54] Teddy Surya Gunawan, Siti Amalina Mohammad Hanafiah, Mira Kartiwi, Nanang Ismail, Nor Farahidah Za'bah, and Anis Nurashikin Nordin. Development of photo forensics algorithm by detecting photoshop manipulation using error level analysis. In *CVPR*, 2017.

[55] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.

[56] Mark Harris. How to optimize data transfers in cuda c/c++. https://developer.nvidia.com/blog/how-optimize-data-transfers-cuda-cc/, 2012.

[57] Mark Harris. Gpu pro tip: Cuda 7 streams simplify concurrency. https://devblogs.nvidia.com/gpu-pro-tip-cuda-7-streams-simplify-concurrency/, 2015.

[58] Mark Harris. Gpu pro tip: Cuda 7 streams simplify concurrency. https://devblogs.nvidia.com/gpu-pro-tip-cuda-7-streams-simplify-concurrency/, 2015.

[59] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[60] Yu-Feng Hsu and Shih-Fu Chang. Detecting image splicing using geometry invariants and camera characteristics consistency. In *2006 IEEE International Conference on Multimedia and Expo*, pages 549–552. IEEE, 2006.

[61] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[62] N. Huang, J. He, and N. Zhu. A novel method for detecting image forgery based on convolutional neural network. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1702–1705, Aug 2018.

[63] Asmaa Ibrahim, Ayat G. Lashen, Ayaka Katayama, Raluca Mihai, Graham Ball, Michael S. Toss, and Emad A. Rakha. Defining the area of mitoses counting in invasive breast cancer using whole slide image. *Modern Pathology*, 2021.

[64] Humayun. Irshad. Automated mitosis detection in histopathology using morphological and multi-channel statistics features. *Journal of Pathology Informatics*, 4(1):10, 2013.

[65] Paul F. Jaeger, Simon A. A. Kohl, Sebastian Bickelhaupt, Fabian Isensee, Tristan Anselm Kuder, Heinz-Peter Schlemmer, and Klaus H. Maier-Hein. Retina u-net: Embarrassingly simple exploitation of segmentation supervision for medical object detection. *CoRR*, abs/1811.08661, 2018.

[66] Stephan Jahn, Markus Plass, and Farid Moinfar. Digital pathology: Advantages, limitations and emerging perspectives. *Journal of Clinical Medicine*, 9:3697, 11 2020.

[67] Abdul Rehman Javed and Zunera Jalil. Byte-level object identification for forensic investigation of digital images. In *2020 International Conference on Cyber Warfare and Security (ICCWS)*, pages 1–4, 2020.

[68] Abdul Rehman Javed and Zunera Jalil. Byte-level object identification for forensic investigation of digital images. In *2020 International Conference on Cyber Warfare and Security (ICCWS)*, pages 1–4, 2020.

[69] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014.

[70] Zhengfen Jiang, Xin Liu, Zhuangzhi Yan, Wenting Gu, and Jiehui Jiang. Improved detection performance in blood cell count by an attention-guided deep learning method. *OSA Continuum*, 4(2):323–333, Feb 2021.

[71] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, Ayush Chaurasia, TaoXie, Liu Changyu, Abhiram V, Laughing, tkianai, yxNONG, Adam Hogan, lorenzomammana, AlexWang1900, Jan Hajek, Laurentiu Diaconu, Marc, Yonghye Kwon, oleg, wanghaoyang0106, Yann Defretin, Aditya Lohia, ml5ah, Ben Milanko, Benjamin

Fineran, Daniel Khromov, Ding Yiwei, Doug, Durgesh, and Francisco Ingham. ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations, April 2021.

[72] Glenn Jocher, Alex Stoken, Ayush Chaurasia, Jirka Borovec, NanoCode012, TaoXie, Yonghye Kwon, Kalen Michael, Liu Changyu, Jiacong Fang, Abhiram V, Laughing, tkianai, yxNONG, Piotr Skalski, Adam Hogan, Jebastin Nadar, imyhxy, Lorenzo Mammana, AlexWang1900, Cristi Fati, Diego Montes, Jan Hajek, Laurentiu Diaconu, Mai Thanh Minh, Marc, albinxavi, fatih, oleg, and wanghaoyang0106. ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support, October 2021.

[73] Michael Kane, John Emerson, and Stephen Weston. Scalable strategies for computing with massive data. *Journal of Statistical Software, Articles*, 55(14):1–19, 2013.

[74] Abhishek Kashyap, Rajesh Singh Parmar, M Agarwal, and Hari Gupta. An evaluation of digital image forgery detection approaches. *International Journal of Applied Engineering Research*, 12:4747–4758, 03 2017.

[75] K. Khuspe and V. Mane. Robust image forgery localization and recognition in copy-move using bag of features and svm. In *2015 International Conference on Communication, Information Computing Technology (ICCICT)*, pages 1–5, Jan 2015.

[76] David Kim, Liron Pantanowitz, Peter Schüffler, DigVijay Yarlagadda, Orly Ardon, VictorE Reuter, Meera Hameed, DavidS Klimstra, and MatthewG Hanna. (re) defining the high-power field for digital pathology. *Journal of Pathology Informatics*, 11:33, 10 2020.

[77] Daisuke Komura and Shumpei Ishikawa. Machine learning methods for histopathological image analysis. *Computational and Structural Biotechnology Journal*, 16:34 – 42, 2018.

[78] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[79] Leeor Langer, Yoav Binenbaum, Leon Gugel, Moran Amit, Ziv Gil, and Shai Dekel. Computer-aided diagnostics in digital pathology: automated evaluation of early-phase pancreatic cancer in mice. *International journal of computer assisted radiology and surgery*, 10, 10 2014.

[80] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[81] Kyong-Ha Lee, Yoon-Joon Lee, Hyunsik Choi, Yon Dohn Chung, and Bongki Moon. Parallel data processing with mapreduce: A survey. *SIGMOD Rec.*, 40(4):11–20, January 2012.

[82] Andréanne Lemay. Kidney recognition in ct using yolov3, 2019.

[83] Chao Li, Xinggang Wang, Wenyu Liu, and Longin Latecki. Deepmitosis: Mitosis detection via deep detection, verication and segmentation networks. *Medical Image Analysis*, 45, 01 2018.

[84] Chao Li, Xinggang Wang, Wenyu Liu, and Longin Jan Latecki. Deepmitosis: Mitosis detection via deep detection, verification and segmentation networks. *Medical Image Analysis*, 45:121 – 133, 2018.

[85] Chao Li, Xinggang Wang, Wenyu Liu, and Longin Jan Latecki. Deepmitosis: Mitosis detection via deep detection, verification and segmentation networks. *Medical Image Analysis*, 45:121 – 133, 2018.

[86] Chao Li, Xinggang Wang, Wenyu Liu, Longin Jan Latecki, Bo Wang, and Junzhou Huang. Weakly supervised mitosis detection in breast histopathology images using concentric loss. *Medical Image Analysis*, 53:165 – 178, 2019.

[87] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *CoRR*, abs/1907.09693, 2019.

[88] Yuguang Li, Ezgi Mercan, Stevan Knezevitch, Joann G. Elmore, and Linda G. Shapiro. Efficient and accurate mitosis detection - a lightweight rcnn approach. In *ICPRAM*, 2018.

[89] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[90] Hwei-Jen Lin, Chun-Wei Wang, and Yang-Ta Kao. Fast copy-move forgery detection. *WSEAS Trans. Sig. Proc.*, 5(5):188–197, May 2009.

[91] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[92] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[93] Tsung-Yu Lin, Aruni Roy Chowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1449–1457, Washington, DC, USA, 2015. IEEE Computer Society.

[94] Bo Liu, Chi-Man Pun, and Xiaochen Yuan. Digital image forgery detection using jpeg features and local noise discrepancies. *TheScientificWorldJournal*, 2014:230425, 03 2014.

[95] Jun Liu and Xuewei Wang. Tomato diseases and pests detection based on improved yolo v3 convolutional neural network. *Frontiers in Plant Science*, 11:898, 2020.

[96] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. *CoRR*, abs/1803.01534, 2018.

[97] Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015.

[98] Xi Lu, Zejun You, Miaomiao Sun, Jing Wu, and Zhihong Zhang. Breast cancer mitotic cell detection using cascade convolutional neural network with u-net. *Mathematical Biosciences and Engineering*, 18:673–695, 04 2021.

[99] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[100] Y. P. Mack. Local properties of *k*-NN regression estimates. *j-SIAM-J-ALG-DISC-METH*, 2(3):311–323, September 1981.

[101] Lester W. Mackey, Ameet Talwalkar, and Michael I. Jordan. Divide-and-conquer matrix factorization. In *NIPS*, pages 1134–1142, 2011.

[102] Babak Mahdian and Stanislav Saic. Using noise inconsistencies for blind image forensics. *Image Vision Comput.*, 27(10):1497–1503, September 2009.

[103] N. Matloff. *Statistical Regression and Classification: From Linear Models to Machine Learning*. CRC Press, 2017.

[104] N. Matloff. *Programming on Parallel Machines*. 2018.

[105] N. Matloff. *A Tour of Recommender Systems*. 2018.

[106] Norm Matloff et al. *partools*, 2015.

[107] Norman Matloff. Software Alchemy: Turning Complex Statistical Computations into Embarassingly–Parallel Ones. *Journal of Statistical Software*, 71(4):1–15, 2016.

[108] A. H. Mir, M. Hanmandlu, and S. N. Tandon. Texture analysis of ct images. *IEEE Engineering in Medicine and Biology Magazine*, 14(6):781–786, Nov 1995.

[109] Minati Mishra and Munesh Adhikary. Digital image tamper detection techniques - a comprehensive study. 06 2013.

[110] Zahra Moghaddasi, Hamid Jalab, Rafidah Md Noor, and Sr Aghabozorgi. Improving rlrn image splicing detection with the use of pca and kernel pca. *TheScientificWorldJournal*, 2014:606570, 09 2014.

[111] Tian-Tsong Ng, Shih-Fu Chang, and Q Sun. A data set of authentic and spliced image blocks. *Columbia University, ADVENT Technical Report*, pages 203–2004, 2004.

[112] Thanh Nguyen, Quang Tran, Nghia Nguyen, and Ha Quy Nguyen. Detection and segmentation of endoscopic artefacts and diseases using deep architectures, 04 2020.

[113] Thanh Nhan Nguyen, Quang Dat Tran, Trung Nghia Nguyen, and Quy Ha Nguyen. Detection and segmentation of endoscopic artefacts and diseases using deep architectures. *medRxiv*, 2020.

[114] Xipeng Pan, Yinghua Lu, Rushi Lan, Zhenbing Liu, Zujun Qin, Huadeng Wang, and Zaiyi Liu. Mitosis detection techniques in h&e stained breast cancer pathological images: A comprehensive review. *Computers & Electrical Engineering*, 91:107038, 2021.

[115] Yuechao Pan, Yangzihao Wang, Yuduo Wu, Carl Yang, and John D. Owens. Multi-gpu graph analytics. *CoRR*, abs/1504.04804, 2015.

[116] Angshuman Paul and Dipti Prasad Mukherjee. Mitosis detection for invasive breast cancer grading in histopathological images. *IEEE Transactions on Image Processing*, 24(11):4041–4054, 2015.

[117] Munish Puri, ShelleyB Hoover, StephenM Hewitt, Bih-Rong Wei, Hibret Adissu, Charles Halsey, Jessica Beck, Charles Bradley, SarahD Cramer, AmyC Durham, D. Esplin, Chad Frank, LTiffany Lyle, LawrenceD McGill, MelissaD Sánchez, PaulaA Schaffer, RyanP Traslavina, Elizabeth Buza, HowardH Yang, and RMark Simpson. Automated computational detection, quantitation, and mapping of mitosis in whole-slide images for clinically actionable surgical pathology decision support. *Journal of Pathology Informatics*, 10:4, 02 2019.

[118] Ghulam Qadir, Syamsul Yahaya, and Anthony Tung Shuen Ho. Surrey university library for forensic analysis (sulfa) of video content. 2012.

[119] Yixuan Qiu, Chih-Jen Lin, Yu-Chin Juan, Wei-Sheng Chin, Yong Zhuang, Bo-Wen Yuan, Meng-Yuan Yang, and other contributors. See file AUTHORS for details. *recosystem: Recommender System using Matrix Factorization*, 2017. R package version 0.4.2.

[120] Pooja Rajkumar and Norman Matloff. Rectools: an advanced recommender system. *useR!*, 2018.

[121] Emad Rakha, Jorge Reis-Filho, Frederick Baehner, David Dabbs, Thomas Decker, Vincenzo Eusebi, Stephen Fox, Shu Ichihara, Jocelyne Jacquemier, Sr Lakhani, José Palacios, Andrea Richardson, Stuart Schnitt, Fernando Schmitt, Puay-Hoon Tan, Gary Tse, Sunil Badve, and Ian Ellis. Breast cancer prognostic classification in the molecular era: the role of histological grade. *Breast cancer research : BCR*, 12:207, 07 2010.

[122] Brijesha D. Rao and Mukesh M. Goswami. A comprehensive study of features used for brian tumor detection and segmentation from mr images. In *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–6, 2017.

[123] Siddhant Rao. MITOS-RCNN: A novel approach to mitotic figure detection in breast cancer histopathology images using region based convolutional neural networks. *CoRR*, abs/1807.01788, 2018.

[124] Punam Sunil Raskar and Sanjeevani Kiran Shah. Real time object-based video forgery detection using yolo (v2). *Forensic Science International*, 327:110979, 2021.

[125] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[126] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[127] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[128] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[129] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[130] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, June 2017.

[131] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[132] Minsoo Rhu, Natalia Gimelshein, Jason Clemons, Arslan Zulfiqar, and Stephen W. Keckler. vdnn: Virtualized deep neural networks for scalable, memory-efficient neural network design, 10 2016.

[133] Mark Robins. Ai and hpc: Challenges and opportunities. Keynote, ICPP 2018, 2018.

[134] Emanuel Rodriguez. Mapreduce on airline data.

[135] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[136] L. Roux. Mitos-atypia-14 - grand challenge, 2014.

[137] Ludovic. Roux, Daniel. Racoceanu, Nicolas. Loménie, Maria. Kulikova, Humayun. Irshad, Jacques. Klossa, Frédérique. Capron, Catherine. Genestie, Gilles. Naour, and Metin. Gurcan. Mitosis detection in breast cancer histological images An ICPR 2012 contest. *Journal of Pathology Informatics*, 4(1):8, 2013.

[138] Sindhu Ramachandran S., Jose George, Shibon Skaria, and Varun V. V. Using YOLO based deep learning network for real time detection and localization of lung nodules from low dose CT scans. In Nicholas Petrick and Kensaku Mori, editors, *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575, pages 347 – 355. International Society for Optics and Photonics, SPIE, 2018.

[139] Sindhu Ramachandran S., Jose George, Shibon Skaria, and Varun V. V. Using YOLO based deep learning network for real time detection and localization of lung nodules from low dose CT scans. In Nicholas Petrick and Kensaku Mori, editors, *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575, pages 347 – 355. International Society for Optics and Photonics, SPIE, 2018.

[140] Maytal Saar-Tsechansky and Foster J. Provost. Handling missing values when applying classification models. *Journal of Machine Learning Research*, 8:1623–1657, 2007.

[141] K. Sabeena Beevi, Madhu S. Nair, and G.R. Bindu. Automatic mitosis detection in breast histopathology images using convolutional neural network based deep transfer learning. *Biocybernetics and Biomedical Engineering*, 39(1):214–223, 2019.

[142] Monjoy Saha, Chandan Chakraborty, and Daniel Racoceanu. Efficient deep learning model for mitosis detection using breast histopathology images. *Computerized Medical Imaging and Graphics*, 64, 12 2017.

[143] Monjoy Saha, Chandan Chakraborty, and Daniel Racoceanu. Efficient deep learning model for mitosis detection using breast histopathology images. *Computerized Medical Imaging and Graphics*, 64, 12 2017.

[144] Meriem Sebai, Xinggang Wang, and Tianjiang Wang. Maskmitosis: a deep learning framework for fully supervised, weakly supervised, and unsupervised mitosis detection in histopathology images. *Medical & Biological Engineering & Computing*, 58, 05 2020.

[145] Nida Shahid, Tim Rappon, and Whitney Berta. Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PLOS ONE*, 14:1–22, 02 2019.

[146] Varsha Sharma and Swati Jha. Image forgery and it's detection technique: A review. 2016.

[147] Alex Shonenkov. WBF approach for ensemble. `https://www.kaggle.com/shonenkov/wbf-approach-for-ensemble/`, 2020.

[148] Anabia Sohail, Asifullah Khan, Noorul Wahab, Aneela Zameer, and Saranjam Khan. A multi-phase deep cnn based mitosis detection framework for breast cancer histopathological images. *Scientific Reports*, 11:6215, 03 2021.

[149] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: ensembling boxes for object detection models, 2020.

[150] Kumar Sunil, Desai Jagan, and Mukherjee Shaktidev. Dct-pca based method for copy-move forgery detection. In Suresh Chandra Satapathy, P. S. Avadhani, Siba K. Udgata, and Sadasivuni Lakshminarayana, editors, *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India- Vol II*, pages 577–583, Cham, 2014. Springer International Publishing.

[151] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[152] Kazuhiro Tabata, Naohiro Uraoka, Jamal Benhamida, Matthew Hanna, Sahussapont Sirintrapun, Brandon Gallas, Qi Gong, Rania Aly, Katsura Emoto, Kant Matsuda, Meera Hameed, David Klimstra, and Yukako Yagi. Validation of mitotic cell quantification via microscopy and multiple whole-slide scanners. *Diagnostic Pathology*, 14, 06 2019.

[153] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection, 2019.

[154] Ashkan Tashk, Mohammad Sadegh Helfroush, Habibollah Danyali, and Mojgan Akbarzadeh. An automatic mitosis detection method for breast cancer histopathology slide images based on objective and pixel-wise textural features classification. In *The 5th Conference on Information and Knowledge Technology*, pages 406–410, 2013.

[155] Rahul Thakur and Rajesh Rohilla. Recent advances in digital image manipulation detection techniques: A brief review. *Forensic Science International*, 312:110311, 2020.

[156] Thaína Tosta, Leandro A. Neves, and Marcelo Zanchetta do Nascimento. Segmentation methods of he-stained histological images of lymphoma: A review. *Informatics in Medicine Unlocked*, 9, 05 2017.

[157] Alex Townsend. JPEG: Image compression algorithm. `http://pi.math.cornell.edu/~web6140/TopTenAlgorithms/JPEG.html`, 2020.

[158] Dijana Tralic, Ivan Zupancic, Sonja Grgic, and Mislav Grgic. Comofod -new database for copy-move forgery detection. 09 2013.

[159] Sujatha R. Upadhyaya. Parallel approaches to machine learning—a comprehensive survey. *Journal of Parallel and Distributed Computing*, 73(3):284 – 292, 2013. Models and Algorithms for High-Performance Distributed Data Mining.

[160] Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67, 2011.

[161] Mitko Veta, Yujing J. Heng, Nikolas Stathonikos, Babak Ehteshami Bejnordi, Francisco Beca, Thomas Wollmann, Karl Rohr, Manan A. Shah, Dayong Wang, Mikaël Rousson, Martin Hedlund, David Tellez, Francesco Ciompi, Erwan Zerhouni, David Lanyi, Matheus Palhares Viana, Vassili A. Kovalev, Vitali Liauchuk, Hady Ahmady Phoulady, Talha Qaiser, Simon Graham, Nasir M. Rajpoot, Erik Sjöblom, Jesper Molin, Kyunghyun Paeng, Sangheum Hwang, Sunggyun Park, Zhipeng Jia, Eric I-Chao Chang, Yan Xu, Andrew H. Beck, Paul J. van Diest, and Josien P. W. Pluim. Predicting breast tumor proliferation from whole-slide images: The tupac16 challenge. *Medical Image Analysis*, 54:111–121, 2019.

[162] Mitko Veta, Paul J. van Diest, Mehdi Jiwa, Shaimaa Al-Janabi, and Josien P. W. Pluim. Mitosis counting in breast cancer: Object-level interobserver agreement and comparison to an automatic method. *PLoS ONE*, 11(8), 8 2016.

[163] Mitko Veta, Paul J. van Diest, Stefan M. Willems, Haibo Wang, Anant Madabhushi, Angel Cruz-Roa, Fabio A. González, Anders Boesen Lindbo Larsen, Jacob S. Vestergaard, Anders B. Dahl, Dan C. Ciresan, Jürgen Schmidhuber, Alessandro Giusti, Luca Maria Gambardella, F. Boray Tek, Thomas Walter, Ching-Wei Wang, Satoshi Kondo, Bogdan J. Matuszewski, Frédéric Precioso, Violet Snell, Josef Kittler, Teófilo Emídio de Campos, Adnan Mujahid Khan, Nasir M. Rajpoot, Evdokia Arkoumani, Miangela M. Lacle, Max A. Viergever, and Josien P. W. Pluim. Assessment of algorithms for mitosis detection in breast cancer histopathology images. *CoRR*, abs/1411.5825, 2014.

[164] Anish Singh Walia. How to implement deep learning in r using keras and tensorflow. https://towardsdatascience.com/how-to-implement-deep-learning-in-r-using-keras-and-tensorflow-82d135ae4889, 2017.

[165] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network, 2021.

[166] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13029–13038, June 2021.

[167] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. Cspnet: A new backbone that can enhance learning capability of cnn, 2019.

[168] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks. *CoRR*, abs/2105.04206, 2021.

[169] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks, 2021.

[170] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206*, 2021.

[171] Haibo Wang, Angel Cruz-Roa, Ajay Basavanhally, Hannah Gilmore, Natalie Shih, Mike Feldman, John Tomaszewski, Fabio González, and Anant Madabhushi. Mitosis detection in breast cancer pathology images by combining handcrafted and convolutional neural network features. *Journal of Medical Imaging*, 1:1–8, 12 2014.

[172] Y. Wei, X. Bi, and B. Xiao. C2r net: The coarse to refined network for image forgery detection. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1656–1659, Aug 2018.

[173] Bihan Wen, Ye Zhu, Ramanathan Subramanian, Tian-Tsong Ng, Xuanjing Shen, and Stefan Winkler. Coverage – a novel database for copy-move forgery detection. In *IEEE International Conference on Image processing (ICIP)*, pages 161–165, 2016.

[174] Yue Wu, Wael Abd-Almageed, and P. Natarajan. Busternet: Detecting copy-move image forgery with source/target localization. In *ECCV*, 2018.

[175] Ellery Wulczyn, David F. Steiner, Zhaoyang Xu, Apaar Sadhwani, Hongwu Wang, Isabelle Flament-Auvigne, Craig H. Mermel, Po-Hsuan Cameron Chen, Yun Liu, and Martin C. Stumpe. Deep learning-based survival prediction for multiple cancer types using histopathology images. *PLOS ONE*, 15(6):1–18, 06 2020.

[176] Yanyang Yan, Wenqi Ren, and Xiaochun Cao. Recolored image detection via a deep discriminative model. *IEEE Transactions on Information Forensics and Security*, 14(1):5–17, 2019.

[177] Robin Yancey. *imagefraud*, 2019.

[178] Robin Yancey. ECS171_summer2020. `https://colab.research.google.com/drive/1hOmmt0SMuYAva1X-p6i4VnlBH4TNVFP-`, 2020.

[179] Robin Elizabeth Yancey. Multi-stream faster rcnn for mitosis counting in breast cancer images, 2020.

[180] Shangjie Yao, Yaowu Chen, Xiang Tian, Rongxin Jiang, and Shuhao Ma. An improved algorithm for detecting pneumonia based on yolov3. *Applied Sciences*, 10:1818, 03 2020.

[181] S. Ye, Q. Sun, and E. Chang. Detecting digital image forgeries by measuring inconsistencies of blocking artifact. In *2007 IEEE International Conference on Multimedia and Expo*, pages 12–15, July 2007.

[182] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019.

[183] Yuan Zhang, Jin Chen, and Xianzhu Pan. Multi-feature fusion of deep networks for mitosis segmentation in histological images. *International Journal of Imaging Systems and Technology*, 31, 06 2021.

[184] Xudong Zhao, Jianhua Li, Shenghong Li, and Shilin Wang. Detecting digital image splicing in chroma spaces. In Hyoung-Joong Kim, Yun Qing Shi, and Mauro Barni, editors, *Digital Watermarking*, pages 12–22, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[185] Lilei Zheng, Ying Zhang, and Vrizlynn L.L. Thing. A survey on image tampering and its detection in real-world photos. *Journal of Visual Communication and Image Representation*, 58:380–399, 2019.

[186] Peng Zhou, Xintong Han, Vlad I. Morariu, and Larry S. Davis. Learning rich features for image manipulation detection. In *CVPR*, 2018.

[187] Peng Zhou, Xintong Han, Vlad I Morariu, and Larry S Davis. Learning rich features for image manipulation detection. In *CVPR*, 2018.

[188] Peng Zhou, Xintong Han, Vlad I. Morariu, and Larry S. Davis. Learning rich features for image manipulation detection. *CoRR*, abs/1805.04953, 2018.