Detecting Post Editing of Multimedia Images using Transfer Learning and Fine Tuning

SIMON JONKER, MALTHE JELSTRUP, WEIZHI MENG, and BROOKE LAMPE, Technical University of Denmark, Denmark

In the domain of general image forgery detection, a myriad of different classification solutions have been developed to distinguish a "tampered" image from a "pristine" image. In this work, we aim to develop a new method to tackle the problem of binary image forgery detection. Our approach builds upon the extensive training that state-of-the-art image classification models have undergone on regular images from the ImageNet dataset, and transfers that knowledge to the image forgery detection space. By leveraging transfer learning and fine tuning, we can fit state-of-the-art image classification models to the forgery detection task. We train the models on a diverse and evenly distributed image forgery dataset. With five models—EfficientNetB0, VGG16, Xception, ResNet50V2, and NASNet-Large—we transferred and adapted pre-trained knowledge from ImageNet to the forgery detection task. Each model was fitted, fine-tuned, and evaluated according to a set of performance metrics. Our evaluation demonstrated the efficacy of large-scale image classification models—paired with transfer learning and fine tuning—at detecting image forgeries. When pitted against a previously unseen dataset, the best-performing model of EfficientNetB0 could achieve an accuracy rate of nearly 89.7%.

CCS Concepts: • **Security and privacy → Intrusion detection systems**; *Distributed systems security*; *Mobile and wireless security*; *Denial-of-service attacks;*

Additional Key Words and Phrases: Multimedia data integrity, image forgery, fake news, post editing, fine tuning, transfer learning

ACM Reference format:
Simon Jonker, Malthe Jelstrup, Weizhi Meng, and Brooke Lampe. 2024. Detecting Post Editing of Multimedia Images using Transfer Learning and Fine Tuning. *ACM Trans. Multimedia Comput. Commun. Appl.* 20, 6, Article 154 (March 2024), 22 pages.
https://doi.org/10.1145/3633284

1 INTRODUCTION

For multimedia platforms, image forgery has become increasingly accessible through more complex editors, better automated image overlaying, and coding-wise image editing [19, 37]. As such, the ability to differentiate authentic images from forged images has become ever more necessary [32]. This goes for both private individuals who may be struggling with finding the genuine Twitter account for their favorite celebrity and in the courtroom where authenticity of images

Authors' addresses: S. Jonker, M. Jelstrup, W. Meng (Corresponding author), and B. Lampe, Technical University of Denmark, DTU Compute, Richard Petersens Plads, Lyngby 2800, Denmark; e-mails: {s184297, s184291}@student.dtu.dk; {weme, blam}@dtu.dk.

1551-6857/2024/03-ART154 $15.00
https://doi.org/10.1145/3633284

and detection of forgery would be an immaculate tool for making sure the right culprit ends up behind bars. Lots of sensitive information is being digitized and the validation and integrity of said information is imperative [5]. The above naturally calls urgently for ways of detecting these falsifications, preferably with the largest degree of automation and smallest degree of manual labor, given the ever increasing amount of information in need of verification [34] and retrieval [12].

The expression "Fake News" has become more prominent and has previously been seen linked to everything from individuals spreading rumours via social media to state sponsored allegations via corrupt media. For example, a TikTok video was posted in August 2022, spreading that Disney World was going to lower the drinking age to 18 [1]. The need for authenticity often arises when reading articles with headlines meant to inspire outrage. One of the ways for these rumors to circumvent that issue is by forging images to make it look like someone or something is behaving according to the story. Although the term "Fake News" is commonly associated with the election campaign of former U.S. President Donald Trump, it is not a new expression—nor is it going away [2].

While completely validating all types of information that can be put in writing would not be possible via computations, perhaps a validation of potentially forged images are. Not only does imagery have a remarkable impact on various areas such as criminal investigation, surveillance systems, intelligence systems, legal services and insurance claim, but also a direct effect of the increased demand for detecting forged images shows itself an important topic in both academia and industry [20].

Motivations. The most original approaches to solving the image forgery detection problem were the development of a model that would utilize outlier-detection in different color spectra as well as different image types. Given its relatively simple setup, these original approaches were intended as a proof-of-concept project, in which the usage of image editors in Python would facilitate concept creation. Simple photo edits were conducted to enable subsequent attempts at detection. Likewise, an even simpler model—intended to be used as comparison baseline for later results—was implemented.

Although the potential for problems with time consumption in training and application of the models was well-known, the expectation was that the detection would be good and at least decent enough to provide some significant results [16, 43]. With this in mind and a general idea of proof-of-concept, there is a need to consider the following aspects: a) the extra time it would take to train and predict on the model would be outweighed by the simple approach, b) few pictures needed (no large dataset), c) a utilization of the vast amounts of data in each picture, d) good visualization provided to highlight and illustrate the relevant findings. Time was also set aside to be able to optimize the models and make them as efficient as possible. This would be achieved by GPU usage and Keras, even going as far as implementations for server side serving on the models created. This could have potentially provided the speed needed for simple image forgery detection with the help of outlier-detection. However, simple testing on a normal CPU proved to be inadequate and insufficient for achieving desired detection rate, so even accounting for the time spent, there was no reason to continue this kind of attempt.

Likewise, for the purpose of optimization, the conversion module `Hummingbird-ml` was utilized to convert the common Scikit-Learn algorithms to be faster but equal implementation in Pytorch. This also gave the possibility of Keras and GPU usage but as previously mentioned, the idea never made it that far. However, the conversion was still valid, as it still utilized the more streamlined algorithms and built blocks provided by the Pytorch package, thereby making the computations much faster.

The computations fed to the models were simple mathematical differences of various flavors. Multiple approaches were explored, such as increasing the averages that were used in detection.

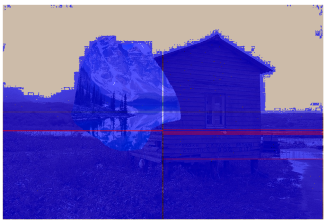

Fig. 1. Inadequate results of attempted outlier detection.

"Increasing the averages" means that instead of adding the simple difference of two pixels to the data, the computation would be an average of the 10 differences of the 10 preceding pixels. In this manner, we can eliminate single points of failure. Unfortunately, this approach proved inefficient since the **True Positives** (**TPs**) were diminished by the calculated averages.

After running several detection types, with many different models and data, it becomes apparent that a more complex approach and algorithms are needed. Relying solely on outlier detection would be inadequate, and a more complex structure for detection was needed such as a neural network. As is apparent in the remainder of this work, changing the approach proved to be the right choice. As this was a learning process and not the basis for any of the conclusions made in this work, a single image containing the best performing model is displayed below for reference (see Figure 1). It is the Isolation Forest run on a simple RGB image, which, as we can see, did not provide any valuable information.

Contributions: This work focuses on the detection of forged imagery with the help from machine-learning models, and, more specifically, the usage of transfer learning and neural networks. Tensorflow and the Keras interface are our tools of choice for modeling the detection algorithms, and we utilize different error measurements to determine the most effective model. Our work attempts to treat image forgery detection as a general image classification problem and therefore to leverage powerful pre-trained state-of-the-art models—popular both in the field and in the literature—in an effort to solve it. Our contributions can be summarized as below:

- We transfer the knowledge of five pre-trained image classification models—EfficientNetB0, VGG16, Xception, ResNet50V2, and NASNet-Large. These models are trained, fitted, fine-tuned, and compared via a set of performance metrics. Each model trains for 100 epochs with most weights non-trainable and the best version of the model advances. This "best" incarnation of the model is then adjusted and trained for an additional 100 epochs with several more trainable parameters. The best-performing incarnation of the fine-tuned model demonstrates the efficacy of image classification models when it comes to differentiating tampered images from pristine images.
- Our evaluation confirms that fine tuning said models has a tremendous impact on model performance, indicating that there are distinct differences between the problems of (1) image classification and (2) tampering detection. We compare both training accuracy and validation accuracy to give an assessment of potential overfitting in spite of the measures we take to reduce this risk (e.g., dropout layers). With the best performing model, EfficientNetB0, we achieved an accuracy of almost 89.7% on an unseen testing set.

Roadmap: The rest of this article is structured as follows: Section 2 describes the related work on image forgery. Section 3 introduces our proposed approach in detail. Section 4 details the implementation and evaluation results. Section 5 discusses open challenges and experimental results. Finally, we conclude this work in Section 6.

Table 1. Summary of Survey Papers on Image Forgery Detection

Year	Survey title
2011 [31]	Forensics Investigations of Multimedia Data: A Review of the State-of-the-Art
2017 [13]	Comparison between image forgery detection algorithms
2019 [10]	Digital Image Forgery Detection Techniques: A Comprehensive Review
2020 [25]	A Comparative Study of Block-Based Copy-Move Forgery Detection Techniques
2021 [28]	A Brief Review on Existing Techniques for Detecting Digital Image Forgery
2021 [17]	Image forgery detection review
2021 [37]	Hierarchical Categorization and Review of Recent Techniques on Image Forgery Detection
2022 [18]	A Review on Digital Image Forgery Detection
2022 [4]	A Comparative Analysis of Image Forgery Detection Techniques
2023 [19]	Toward Deep-Learning-Based Methods in Image Forgery Detection: A Survey

2 RELATED WORK

Image forgeries have led to an explosion of relevant research in recent years, resulting in various surveys. Table 1 provides a summary of survey papers on image forgery detection. This section will briefly introduce related studies on the intersection of image forgery detection via deep learning and transfer learning. Transfer learning is a useful technique (or can be considered as a learning method) for deep learning, by reusing existing pre-trained models to a new problem or domain [45].

Wei et al. [40] aimed to detect faked image or objects on images via rescaling/rotation detection and parameter estimation, based on the relations between the rotation angle and the frequencies. This is because a forged image usually takes rescaling or rotation actions. Chierchia et al. [15] introduced a method of detecting forged images using sensor pattern noise via Bayesian estimation. They also used the Modern convex optimization method to reach a globally optimal solution for estimating **photo-response non-uniformity** (**PRNU**) noise. Pun et al. [14] aimed to detect copy-move forgery actions by using adaptive oversegmentation and feature point matching, which is a combination of block-based and keypoint-based forgery detection approaches. They also introduced a forgery region extraction algorithm to replace the feature points with small superpixels as feature blocks, which can enhance the detection performance.

Mayer and Stamm [9] then introduced a method of image forgery detection via analyzing the inconsistencies of **lateral chromatic aberration** (**LCA**). For this reason, they provided a statistical model that can capture the inconsistency between global and local estimates of LCA. They further posed forgery detection as a hypothesis testing problem and derive the results. Li and Zhou [21] proposed a copy-move forgery detection algorithm via hierarchical feature point matching. They also provided an iterative localization method by exploiting the robustness properties (e.g., dominant orientation) and the color information of each keypoint. Matern et al. [30] focused on image inconsistencies using an analytic model and designed a physics-based forensic detection to characterize 2-D lighting environments of objects. This is because the integral over a gradient field of an object indicates the direction of incident light in the image plane.

Wu et al. [41] focused on the forged images in **online social networks** (**OSNs**) and designed a robust training scheme. They particularly analyzed the noise and decoupled it into predictable noise (caused by operations of OSNs) and unseen noise (the defects of the detector). Gupta et al. [3] evaluated two CNN models for image forgery detection—an untrained CNN and a pre-trained VGG16 CNN. The two models achieved accuracy values of 74.95% and 91.62%, respectively. Then Baviskar et al. [4] developed an 8-layer CNN-based image forgery detection model, in which they evaluated against two pre-trained models: VGG16 and VGG19. For the experimental evaluation, they leveraged the Casia and MICC F2000 datasets. The proposed model achieved an accuracy of 0.95, outperforming both the VGG16 model—0.88, and the VGG19 model—0.90.

Hebbar and Kunte [6] proposed an approach to detect image forgeries that incorporates a pre-trained ResNet50 network—for transfer learning—into the encoder of a DeepLabv3+ architecture. They compared the performance of their proposed approach to five related studies, and they found that their approach significantly outperformed its contemporaries. The F1-scores of the related approaches ranged from 0.20730 to 0.68300, while the proposed approach achieved an F1-score of 0.7510. Cristin et al. [7] devised an image forgery detection scheme that leveraged supervised learning and was trained on both compressed and uncompressed images. They conducted a number of performance analyses in which they varied the learning rate of the activation function—which was either ReLU or Leaky ReLU. Das and Naskar [8] devised a scheme that focuses specifically on the detection of image splicing-type forgeries. They constructed a deep CNN-based model, replacing the initial convolution layers with the pre-trained weights of MobileNetV2 (a CNN that is pre-trained on ImageNet). They reported an accuracy of 93.01%, which surpassed the accuracy scores of the related schemes included in their evaluation and their comparison.

In addition to the image forgery detection schemes, image encryption [39], traditional traffic analysis [24, 27], and collaborative intrusion detection systems [22, 23, 26] can be complementary to enhance the detection scope and protect the overall systems.

3 OUR PROPOSED APPROACH

The proposed method in this article is to use the architecture of established deep learning models in combination with transfer learning and fine tuning in an attempt to uncover which of these is the most suitable for the task of passively detecting image forgeries. For transfer learning and fine tuning purposes, each model's initial weights will be pulled from its training on the ImageNet dataset (https://www.image-net.org/). The ImageNet dataset is a well-known dataset, often mentioned in the literature when it comes to image classification challenges and evaluations. It is very effective when comparing the capabilities of different models. As is the case when using transfer learning, the model weights will be loaded and completely frozen, then the fully connected top layers will be removed and replaced by new ones, after which the model will train and validate on the dataset in hopes of learning the distinction between a pristine and tampered image.

As mentioned, when using transfer learning, the model layers with the exception of the fully connected top layers are frozen, meaning that they are made non-trainable. That is the distinction to fine tuning, in which some or all layers of the model can be unfrozen after loading in weights. Fine tuning provides the model an increased number of trainable parameters and is therefore beneficial whenever, there is a large degree of difference between (1) the current dataset and model and (2) the dataset and model task from which the weights were inherited. This phenomenon can be attributed to an increased number of trainable parameters, which allows the model to refocus on the elements of an image that are relevant to the current task but are not relevant to the "parent" model.

3.1 Dataset

The goal of this sort of classification is to achieve a model that generalizes well on data outside the dataset. To achieve this purpose, one of the most important factors is the dataset itself. For this reason, the dataset used in this article is a unified dataset, composed of a set of publicly available datasets that have been tailored to include only image forgeries of the sort we are attempting to classify—namely, copy, move, and splicing forgeries. Many attempts were made to include datasets not available to the public, in order to further increase the models' generalization power; however, none of those efforts proved fruitful. Therefore, our dataset, henceforth referred to as "Omega" includes images from the following datasets: CASIA v1, CoMoFoD, CG-1050-V1,

Table 2. Breakdown of Unified Dataset–Omega from Each Single Dataset

Dataset	Pristine	Tampered
CASIA v1[1]	0	921
CoMoFoD[2]	48	200
CG-1050-V1[3]	100	381
CG-1050-V2[4]	818	66
COVERAGE[5]	100	100
MICC-F2000[6]	1,300	700
MICC-F220[7]	92	90
Total	2,458	2,458

CG-1050-V2, COVERAGE, MICC-F2000, and MICC-F220. Table 2 shows how many and what kind of images each dataset has provided.

As shown in Table 2, the Omega dataset is composed of equal parts pristine and tampered images, as to not allow for class bias in the models.

3.2 Performance Measures

The models implemented in this article will be evaluated on four performance metrics; accuracy, precision, recall and F1 score. The following will be clarification of what is meant with each of these metrics. Accuracy (A) is intuitively the number of correctly predicted images out of all the predicted images. The precision (P) of the model is the ratio of correctly predicted tampered images out of all the predicted tampered images. Recall (R) measures the ratio of correctly predicted tampered images out of all the tampered images. The models F1 score (F1) is weighted mean of precision and recall. The metrics can be calculated using the Equations (1)–(4).

$$A = \frac{TP + TN}{TP + FP + FN + TN}, \tag{1}$$

$$P = \frac{TP}{TP + FP}, \tag{2}$$

$$R = \frac{TP}{TP + FN}, \tag{3}$$

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}. \tag{4}$$

For these metrics, TP is the number of True Positives, TN is the number of True Negatives, FP is the number of False Positives and FN is the number of False Negatives. The positive prediction is, in this article, a tampered image, whereas a negative prediction is a pristine image. All four metrics range between 0 and 1, with 1 being the ideal and 0 the worst value.

3.3 Tensorflow

Tensorflow (https://www.tensorflow.org/) is a package originally made for Google Brain that has its main applicability in relations to deep-learning algorithms and can be considered sort of the building blocks of algorithms of all kinds. Tensorflow's brilliance partly comes in the utilization of multiple GPUs, which allows for greater efficiency specifically in the training of the models as well as potential prediction.

In this work, the module called Keras will be implemented to enable fast computations using the GPU. Keras is a framework built on top of the Tensorflow package, which allows for great intercommunication between the two. Keras is the number one deep-learning module when it comes to speed, and, time and time again, Keras has demonstrated that its spectacular efficiency does not come at the cost of its integrity. Not only is Keras used by top agencies such as NASA and CERN, it has also been part of the top five teams on Kaggle, thereby proving its simplicity as well as its capability to scale to more complex computations.

3.4 Models Used

In this work, we leveraged five different models. This will allow for a good basis for comparison as well as a larger potential for finding the most optimal model for image forgery detection. This section contains multiple concepts directly related to advanced algorithms regarding testing, usage, creation and efficiency. All five models are highly complex, this is due to the utilization of pre-trained models has been developed by large companies such as Google. The models used have all proven their worth in different avenues and we will discuss why to choose each model as below.

3.4.1 EfficientNet. EfficientNet is a convolutional neural network—also called "CNN" or "ConvNet". As the name implies, CNNs are a sub-category of neural networks and are well known when it comes to image classification problems. They have been applied to all manner of problem domains and industries, from segmentation to medical image analysis. That said, the foundation and general approach of a CNN model lies in its "full connectivity"—that is, the full connectivity of the embedded neurons. As such, CNNs can be applied to many problems beyond the static image analysis, including natural language processing, brain-computer interfaces, and financial time series—to name a few.

The so-called "full connectivity" of CNNs is mainly a result of **multi-layer perceptrons** (**MLPs**), which are utilized in the training phase. Essentially, full connectivity means that every node implemented in every layer contains a neuron, and that neuron is fully connected to the preceding nodes in the previous layer. Each node has a certain weight to which the connection to other nodes in order to make sure that the full connection in theory does not provide any setbacks. MLPs are known to overfit data again as a result of the full connectivity of neurons; however, well-known countermeasures can be taken to address potential overfitting issues. In fact, CNN-type neural networks are referenced as regularized versions of MLPs because various measures have been incorporated into CNNs such that overfitting is kept to a minimum.

CNNs share the common applicability that the models are so vast that scaling of the models when more computational power is added, can improve the model substantially. This is why the base model of EfficientNet, for example, the B0 version, has a much lower detection rate than the successors. The scaling is done by expanding the network. The network can be expanded in multiple ways, two of which involve concrete expansion of the network width and depth—for example, scaling the network. The last of the three ways of expanding the network is using images of higher resolution when training the model. While all three expansions improve the model, what makes EfficientNet one of the best in its field, is the balancing of the three expansions. Making the network substantially wider that deep, the network will be inefficient and vice versa. That is why EfficientNet has implemented coefficients to uniformly and most efficiently scale the model given any specific kind of data. This is what differentiate and significantly improves the EfficientNet from the other algorithms.

As mentioned above, the scaling is done via implemented coefficients to uniformly scale the network. The example given by Tan and Le [46] in relation to the release of the B7 version is to increase N^2 times more computational resources. Then simply increasing the network depth by

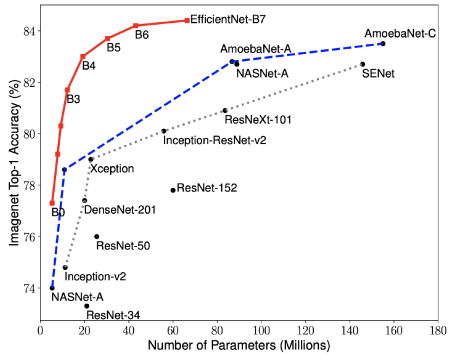

Fig. 2. Model parameters vs. accuracy.

α^2, width by β^2, and image size by γ^2, where $\alpha, \beta, and \gamma$ are constant coefficients determined by a small grid search on the original model, would make sure the expansion of the network was done most efficiently.

A brief explanation of the model—which Efficient builds upon—will help explain why the model is efficient and (as we will demonstrate later) why the model performs the best in terms of detection, resources, and time spent. The EfficientNet predecessor "AutoML MNAS" was developed specifically with the objective of low resource cost as it was developed for mobile devices. This makes for an effective and low cost model foundation to further develop and expand—which is exactly what EfficientNet is.

To look into some of the testing done for this model, a brief introduction to the data will explain why this data is specifically used as a common denominator for testing multiple models. For image classifications and segmentation, the data from ImageNet is absolutely top of the line. This is due to the way the data is structured and the goal of the developers. A large database of so-called "sets of synonyms"—synsets—for many words and word-sets has been constructed and dubbed a "word hierarchy". This word hierarchy comprises almost 100,000 synsets, and the developers of ImageNet strive to provide 1,000 images for each synset. As such, there are a *lot* of images—roughly 1.2 million in total. All images are human verified, controlled, and labeled. This is why testing on such vast data provides an extremely good foundation for comparison.

Figure 2 displays performance results gathered from multiple different sources, based on the application of different models on the ImageNet dataset. Four of the five models utilized in this work are showcased on the graph. This gives a preliminary overview of the strength of each model.

3.4.2 VGG16. Much like the EfficientNet, VGG16 uses a structure with CNNs and inherits the same full connectivity attributes. While the initial thought of a deeper network with similar attributes to the other model sounds like an improvement, one of the first setbacks one would notice of a deeper and more complex network is of course the time consumption. As described in multiple articles [19], this model suffers severely from time issues. This is due to the complex architecture of the model.

The VGG16 model is very complex, which contains multiple algorithm structures such as the `softmax` and other Convolutions as well as an MLP. To explain in full all the algorithms this model utilizes to classify images, one would need more space than this work would allow; however, a quick run-through of the model is as follows: The input RGB image will be input into a stack of convolution layers with very small receptive fields to filter the images. For each *Reduction* in dimension via the filtering, the spatial rotation of the pixels attempts to be preserved by fixing the spatial padding in a 1-pixel convolution layer. This is then used in the max-pooling layers to

achieve the reduction of pixels in the most representative matter. Max-pooling is generally a way to down scale or down sample in this case an image taking the highest value (thus max-pool) as the most representative pixel. After multiple iterations of these layers, the output is an array of representative pixels run through an MLP and classified via the last MLP. The last layer is often a `softmax` layer that aims to classify the input.

VGG16 (https://github.com/ashushekar/VGG16) is an older model introduced in 2014 and was revolutionary at the time, it may now perform worse than current day models given the fields research since then.

3.4.3 Xception. This model has proven to be a more efficient model than VGG16 and is the simplest version of EfficientNet (see Figure 2). However, much like the others, it cannot compete with better-trained EfficientNet models. One of the noticeable differences from this model to the other is that this model has only incorporated the potential for full connectivity in one of the last layers where some of the other models have the Full connectivity as a given part of the end model.

Apart from some of the more commonly seen components of the models such as block normalizations for each of the max-pooling iterations and `ReLU`, the Xception model has its basis on what is called Depthwise Separable Convolution, which in theory should be a faster way of gathering the same results as the commonly used CNN. Common CNN runs on all sides and lengths of the image while the Depthwise Separable Convolution will run on each length and height but will run all data in the depth of the max-pooled data. This is basically a way of simplifying a CNN, making it more lightweight and potentially just as powerful. To make up for the "missing" convolutions seen in common CNN implementations, a so-called "Pointwise" convolution is constructed such that the last dimension is taken into account. Not only did the computations become more lightweight, they also reduce the number of needed computations by a factor of $\frac{1}{Depth}$.

This is one of the models that has the most simplistic architecture even if it provides a similar structure as the others while having eight iterations of the modified convulsion sequences. While developers have aimed for a more simplified way of providing image classification and segmentation, the results are impressive indeed. One such example given by Ganguly et al. [29], in which this exact model was used to detect the so-called "Deep fakes" where faces are swapped.

3.4.4 NASNet-Large. The NASNet-Large or NASNet-A (https://github.com/tensorflow/models/tree/master/research/slim/nets/nasnet) is the largest model that has been implemented in this work. Not only does it have the highest number of parameters, it potentially has the most complex structure as a whole. This is due to the distinctive measures this model takes in order to perform the most efficient detection. It utilizes a whole different setup than any of the other models—it uses Neural Architecture Search (NAS) to search through different types of CNNs (e.g., 2x2 and 5x5.). Since each max-pooling iteration shifts or changes data, it is very likely that different CNNs will provide different results; as such, NAS was implemented to find the most capable CNN method.

While search is needed to iterate, data also needs to be generated for the model to search through. This is done via sampling of the data given and training a child network with a given CNN architecture and calculation of a given accuracy. After all models are sampled, the controller **recurrent neural network** (**RNN**) can pass a probability back of how well data fit (and thus probability in correct prediction) on a small held-out validation set and another model is suggested. All this happens before every CNN is implemented as is the most likely cause of the many parameters needed. Thus, as this is a very heavy model and large in size, it is clear that the computational power needed is indeed quite large. Figure 3 illustrates how the circuits look within the NAS:

The most interesting thing to take from this model is the fact that it makes a good comparison to other models while giving a completely new take of detection methods. The new take on image

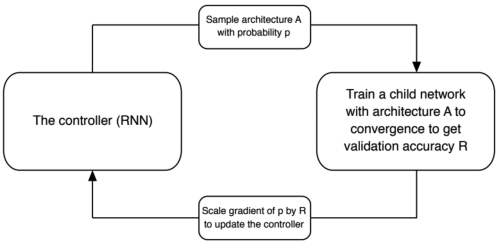

Fig. 3. Overview of neural architecture search (NAS).

classifications is a direct cause of the implementation of the above. The greatest downside of this model is its size, complexity and thereby computational time.

3.4.5 ResNet50V2. ResNet50V2 is, as the name implies, the second version of the ResNet50 and is built upon the theory of Residual Networking. While this network sounds like it is vast in scale, when referencing the 50-block structure it has, of different convolution layers, it actually requires much less computational power. Most of the computational gains may come from the structuring and the overall adaptability the residual networks provide. Residual networks have the benefit of evaluating the layers of the model and provide the adequate "skips" of the convolutions that do not benefit the model in any way. Skipping not only provides computational gains as time does not need to be spent on needless computations, it is also a fundamental part of potentially not overfitting any given data. The general structure has the same philosophy or idea from the VGG nets, which have been covered earlier, it is distinctive in the residual layer added on top.

3.4.6 Choosing Models. When choosing the models in our work, we have to consider a few aspects. The research approach of attempting to treat forgery detection as a classification problem can strongly impact the decision making. EfficientNet was chosen as it is the de facto state-of-the-art model for image classification. It was decided that the EfficientNetB0 would be the version of choice as its parameter count was roughly similar to a few of the other models chosen in this work. The aim of that decision was to have the models be easily comparable. The VGG16 model was chosen as it has previously been a state-of-the-art algorithm for image classification and therefore a valuable model to test the proposed solution. Xception was chosen as it has a great accuracy to parameter count ratio and it was thought ideal to have a model from the "Inception" family. ResNet50V2 was chosen for its unique architecture and approach, it was considered as a good tool to test the proposed solution method. Lastly, the NASNet-Large was chosen as a heavy-hitting model with a large number of parameters—in contrast to the other smaller models used.

3.4.7 General Alterations. As will be discussed in detail in Section 4, we aim to introduce four general approaches in order to optimize all the models for our particular problem. While all four aspects serve some purposes for a given model, this section is oriented toward explaining some of the concepts that will be used at a theoretical level, where the implementation is practically oriented. The following are the four topics:

(1) ReLU. The **Rectified Linear Activation Unit** (`ReLU`) has quickly become the most commonly used activation model for CNNs in particular. There are two reasons for this particular activation model to be preferred than other similar models such as `sigmoid` (which will also be covered) or `tanh`. Once again, the computational power comes to play as the `ReLU` is very fast and lightweight when implemented. This is also why it is common to see this model incorporated multiple times in any of the models tested in this work. It has a simple structure as follows: $f(x) = max(0, x)$. This is a very fast and efficient way to transform the data coming from

the neurons into values higher than or equal to zero, while rectifying the data into a linear problem.

Secondly, the ReLU model provides sparsity with the model implementation. Much like a sparse Matrix where most entities are zero, sparse models with some of the weights being zero usually result in more concise models with less chance of overfitting and parsing data that does not provide value. Due to the transformation of all negative numbers to zero, it is likely to create a sparse network.

The importance of these activation functions is the need for the deep learning models to learn, understand, and train against non-linear data; thus, the model becomes more complex in its structure. Without activation functions, all neural networks would be perceived as a linear regression problem with input X and a set of weights. This is where the concept of "rectifying" is important: it allows the model to train upon data that is not suitable for linear regression, making the model more complex.

(2) Global Average Pooling. Global Average Pooling is a concept that serves similar purposes as the previously mentioned max-pooling. It is a way to downscale vast quantities of data while attempting to make the data as representative as possible between each CNN layer. While the max-pool takes the largest value of any given pool size pre-defined in the model, the global average pooling takes the average of the values in the pool and scales it. While the concept is quite simple, it holds great power and could potentially change the detection rate drastically. According to Lin et al. [11], they managed to get some good results based on their lightweight model structure with the global average pooling implemented in the model.

(3) Dropout layers. Dropout layers are quite simple to understand yet provide a great value for our study of the concepts mentioned in this section. Dropout layers can force the model to train or explore more aspects of the data, as the dropout layer will randomly select neurons in that layer of the model to make sure that the remained neurons are forced to explain more of the classification and by that get trained. This is a measure to reduce the risk of overfitting.

4) Sigmoid. The sigmoid function serves a similar purpose as ReLU, since they are both activation functions and are used to transform data. While ReLU transforms the data into values greater than zero, the sigmoid function returns a given number between 0 and 1, making it perfect for binary activation. The sigmoid is slightly more demanding computation, $sigmoid(x) = 1/(1 + exp(-x))$, but effectively has the same attributes as softmax—which is often applied to the end flow of models when multiple classes are attempted to be detected. It behaves as if the softmax has two elements whereas the second is always zero, making it equally distributed. Values greater than five will tangent toward one while values less than negative five will tangent to zero.

4 IMPLEMENTATION AND EVALUATION

4.1 Model Hyperparameters

For each model utilized in our work, there are choices to be made when implementing them, so called hyperparameters. This subsection will cover each of our decisions and how they affect the performance of our models. The hyperparameters used in training are as follows; the models will train for 100 epochs, the optimizer will be Adam with a learning rate of 0.0001 and a decay rate for the first and second moment of 0.9 and 0.999, the loss function will be binary cross-entropy as it is a binary classification problem and a batch size of 32 will be used. The models themselves will have either a fully connected layer with ReLU activation or a Global Average Pooling layer followed by a dropout layer with value 0.5 to maximize regularization, lastly a fully connected layer of size 1.0 with sigmoid activation is added as the prediction layer, given that this is a binary

```
pre_trained_model = EfficientNetB0(input_shape=(224, 224, 3),
                                   include_top=False,
                                   weights="imagenet",
                                   drop_connect_rate=0.4)

x = layers.GlobalAveragePooling2D(name="avg_pool")(pre_trained_model.output)
x = layers.Dropout(0.5)(x)
x = layers.Dense(1, activation='sigmoid')(x)

EfficientNetB0_model = Model(pre_trained_model.input, x)
```

Fig. 4. Implementation of the EfficientNetB0 model's top layer.

```
pre_trained_model = VGG16(input_shape=(224, 224, 3),
                          weights='imagenet',
                          include_top=False)

x = layers.Flatten()(pre_trained_model.output)
x = layers.Dense(1024, activation="relu")(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(1, activation="sigmoid")(x)

VGG16_model = Model(pre_trained_model.input, x)
```

Fig. 5. Implementation of the VGG16 model's top layer.

```
root_path = os.getcwd()
data_dir = pathlib.Path(root_path + "/Datasets/Omega Dataset")
batch_size = 32
img_height = 224
img_width = 224
seed = 1337
splits = (0.15, 0.15)

train_ds = custom_image_dataset_from_directory(
    data_dir,
    validation_split=splits,
    subset="training",
    seed=seed,
    image_size=(img_height, img_width),
    batch_size=batch_size
)
```

Fig. 6. Implementation of extracting training data from the Omega dataset.

classification problem. Figures 4 and 5 show the implementation of the top layers for EfficientNet and VGG16, respectively.

The Omega dataset will be split into three parts, each with equal parts pristine and tampered images. 70% of the data will be categorized as training data, 15% will be validation data and the remaining 15% will be testing data. The data will be split into these three parts by utilizing a Keras function that has been customized-built to split datasets into three parts instead of the original two. Using the Keras function, we can easily construct the training dataset, as shown in Figure 6, where "custom_image_dataset_from_directory()" refers to the tailor-made function.

4.2 Performance of Transfer Learning

As mentioned earlier, the models will train for 100 epochs, but the weights of all epochs will be saved and the model weights at the epoch with best accuracy on the validation data will be chosen moving forward and be used for future calculations.

The accuracy for the training and validation data over the course of 100 epochs can be seen in Figures 7, 8, 9, and 11, with the best iteration highlighted.

As shown in Figure 7, the EfficientNetB0 model using transfer learning follows a steady learning curve and reaches its peak validation accuracy at epoch 79, with a validation accuracy of 78.7%.

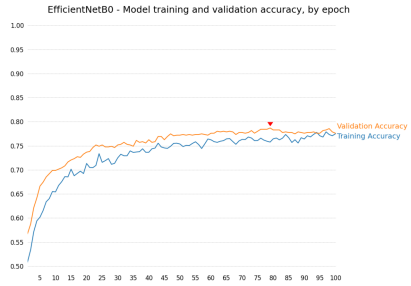

Fig. 7. EfficientNetB0 accuracy performance during 100 epochs of training.

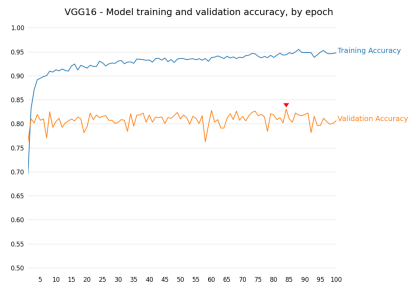

Fig. 8. VGG16 accuracy performance during 100 epochs of transfer learning.

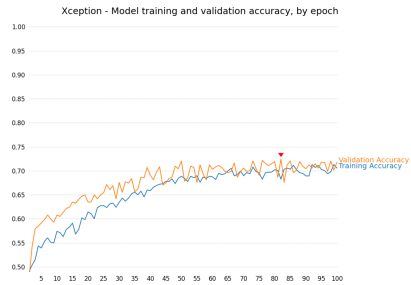

Fig. 9. Xception accuracy performance during 100 epochs of transfer learning.

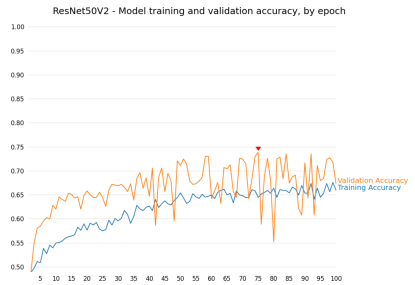

Fig. 10. ResNet50V2 accuracy performance during 100 epochs of transfer learning.

The model achieves this while only having 1,281 trainable parameters. It is additionally clear to see that through transfer learning, the model is unable to reach a high degree of accuracy—either on the training data or the validation data. This indicates that either the problem is too complex for the model, the ImageNet data is too different from the data in the Omega dataset and therefore the inherited weights cannot explain the differences well enough, or that there are simply too few trainable parameters to classify the images effectively.

VGG16 is the model with the most trainable parameters during transfer learning, as it has roughly 25.7 million trainable parameters. As shown in Figure 8, the accuracy on the validation data reaches its peak at epoch 84 with an accuracy of 83.0%. That accuracy is 4.3% greater than the accuracy of EfficientNetB0. However, it is interesting to note that VGG16 has a total amount of trainable and non-trainable parameters of roughly 40.4 million and EfficientNetB0 has 4.1 million. Even more interestingly for transfer learning, VGG16 has about 64% of its parameters trainable, whereas the EfficientNetB0 model has only 0.03% of its parameters trainable. The effects of this are also apparent in Figure 8, where it is clear to see that the VGG16 model experienced significant overfitting very quickly in the training period.

The accuracy of the Xception model, as shown in Figure 9, follows a relatively clean learning curve and shows no acute signs of overfitting. The lack of overfitting shows signs that it might perform even better under fine tuning—as the model has not fit to noise in the data. Therefore, a few more parameter tweaks could help describe the differences in the data better. The Xception model's validation accuracy peaks at epoch 82 with an accuracy of 72.5%. At present, this level of accuracy is not impressive, yet it is important to consider that the Xception model is the smallest of all the models in terms of total parameters: that is, it contains only 20.8 million parameters—and only 2,049 of them are trainable during transfer learning.

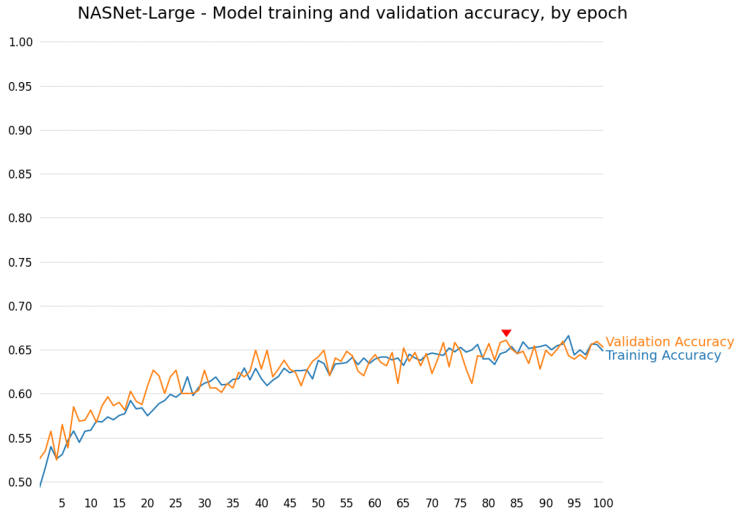

Fig. 11. NASNet-Large accuracy performance during 100 epochs of transfer learning.

The ResNet50V2 is similar in size and proportions to the above-mentioned Xception, yet ResNet50V2 operates in a much-different manner—as described earlier. The result of this different architecture when training and validating on the Omega dataset is shown in Figure 10. When inspecting the figure, the difference in smoothness on the training and validation accuracy curve is apparent—as well as the massive swings in performance on the validation data. It seems to imply that the model has overfit on some element of noise in the training data that results in what resembles coin flips in the accuracy on the validation data. As the accuracy metric tracks only the successfully classified images, and not how far from predicting the class correctly the model was. These large swings indicate that the model does not actually understand much of the dataset and is blindly guessing for the most part. Due to this, the validation accuracy peaks at 73.8% at the 75th epoch.

Figure 11 shows the training and validation accuracy for the NASNet-Large model. It is immediately apparent that the model with its current parameters stagnates around 65% accuracy—more than 10% lower than the accuracy of EfficientNetB0. This is alarming as NASNet-Large has 4,033 trainable parameters, when using transfer learning in this way, which is more than three times as many trainable parameters as the EfficientNetB0 model. Additionally, it is the largest of all tested models by a considerable margin: the total number of parameters for the NASNet-Large model is roughly 84.9 million, more than twice the number of the second largest model. The NASNet-Large model reaches its peak accuracy—66.1%—at epoch 83.

Table 3 provides an overview of each model's performance during its best-performing epoch as well as a breakdown of each model's parameters. From the table, it is immediately apparent that EfficientNetB0 contains by far the fewest parameters, yet it performs very well regardless. VGG16, however, is the best-performing model when it comes to transfer learning—though it is just barely ahead of EfficientNetB0. Our evaluation indicates that NASNet-Large performs significantly worse than the other four models.

4.3 Performance of Fine Tuning

This section covers the fine tuning of the five models. In practice, to effectuate fine tuning, we begin with the incarnation of the model that achieved peak performance during transfer learning.

Table 3. Transfer Learning Model Performance and Parameter Overview

		Accuracy		Parameters			
	Best Epoch	Validation	Training	Trainable	Non-trainable	Total	Trainable Share
EfficientNetB0	79	78.7%	75.8%	1281	4.1 million	4.1 million	0.03%
VGG16	84	83.0%	94.4%	25.7 million	14.7 million	40.4 million	63.6%
Xception	82	72.5%	68.2%	2049	20.9 million	20.9 million	0.01%
ResNet50V2	75	73.8%	64.5%	2049	23.6 million	23.6 million	0.009%
NASNet-Large	83	66.1%	64.8%	4033	84.9 million	84.9 million	0.005%

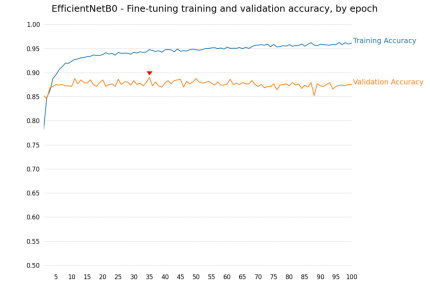

Fig. 12. EfficientNetB0 accuracy performance during 100 epochs of fine tuning.

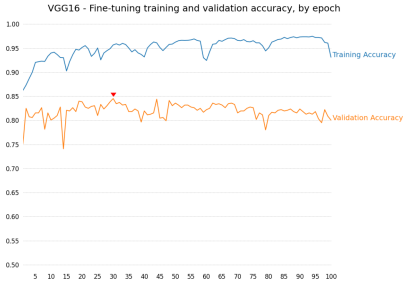

Fig. 13. VGG16 accuracy performance during 100 epochs of fine tuning.

Starting from the top of the model, we select a pre-determined number of layers, which will be made trainable (the number of layers varies by model). Then, we train the model again for 100 epochs. In effect, the model is given a second chance to fit itself to the data. This "second chance" carries an increased risk of overfitting, but it also allows the model to better explain the differences between the images—and thereby better classify them. As earlier mentioned, this phenomenon can be beneficial either (1) if the new dataset differs greatly from the original dataset (in this case, the ImageNet dataset) or (2) if the new classification problem is significantly different from the original classification problem—again, in this case, the ImageNet challenge.

The starting point for fine tuning the EfficientNetB0 model was epoch 79, with a validation accuracy of 78.7%, so beating that is the goal. This goal was more than reached by making an additional 75 layers trainable, which corresponded to two blocks of convolution layers. Thus, the number of trainable parameters increased by 3.1 million, up from a mere number of 1,281. With this change, an immediate difference in performance can be noticed in Figure 12. Within just a few epochs, the validation accuracy reaches a steady plateau between 85 and 90% peaking at exactly 89.0% in epoch 35. After this point, the training accuracy continues to gradually increase, while the validation accuracy starts a slow decline—this is indicative of a model overfitting to its training data. The model experiences an increase in peak performance of 10% from transfer learning to fine tuning.

For the VGG16 model, as shown in Figure 8, the performance increase is extremely minimal: 1.5%. Performance was 83.0% during transfer learning, which increased to 84.5% during fine tuning. This peak during fine tuning occurs at the 30th epoch. The lack of improvement is not surprising, given that the transfer learning performance already showed significant signs of overfitting. The VGG16 model was fine-tuned by unfreezing four additional layers representing the top convolution block and max-pooling layer, which resulted in an additional 7.1 million trainable parameters. This performance evaluation shows that the VGG16 model, on this particular dataset and problem, does not in any meaningful way benefit from fine tuning.

Xception is the model that sees the largest gain from fine tuning in this instance. As shown in Figure 14, the peak accuracy during transfer learning was 72.3%, whereas when fine tuning

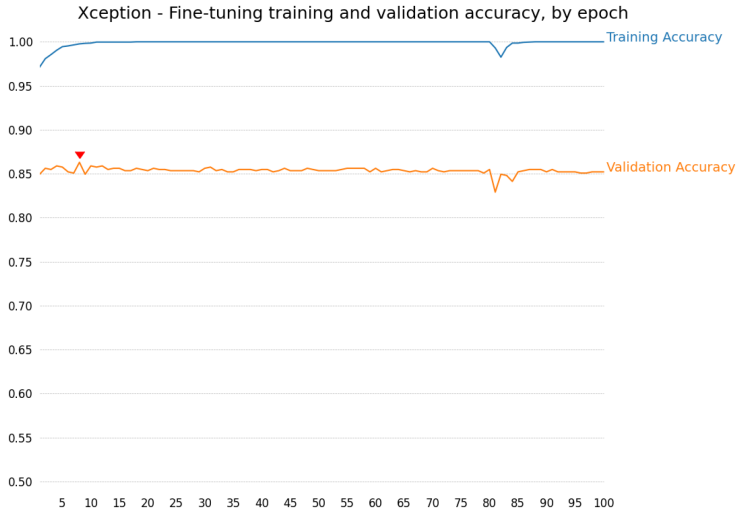

Fig. 14. Xception accuracy performance during 100 epochs of fine tuning.

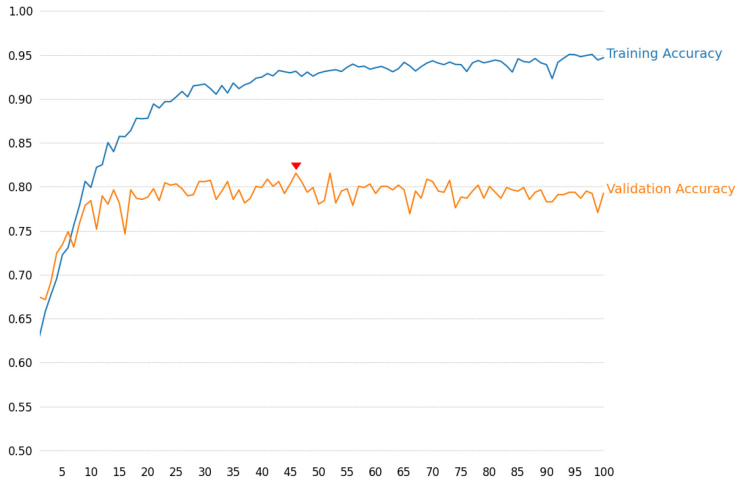

Fig. 15. ResNet50V2 accuracy performance during 100 epochs of fine tuning.

the performance peaks at epoch 8 with an accuracy of 86.3%, an increase of 14%. It was identified during the transfer learning process that Xception was the model with the most room to grow, as it had a relatively low peak accuracy and showed no signs of overfitting. The increased performance, however, was not enough to rival the performance of the EfficientNetB0 model at 89% accuracy, as the model does experience quite significant overfitting during fine tuning. The fine tuning on the Xception model is achieved by unfreezing the top two blocks of the model, which is equivalent to 16 layers or an additional 6.8 million trainable parameters.

The performance increase for the ResNet50V2, as shown in Figure 15, is not only an increase in accuracy, but also a significant increase in model stability. For fine tuning the top convolution block, in other words an additional 13 layers, has been unfrozen, resulting in an increase in trainable parameters of 4.5 million from the original 2,049. This increase in model capabilities lets it more accurately distinguish between pristine and tampered images, to the point where it no

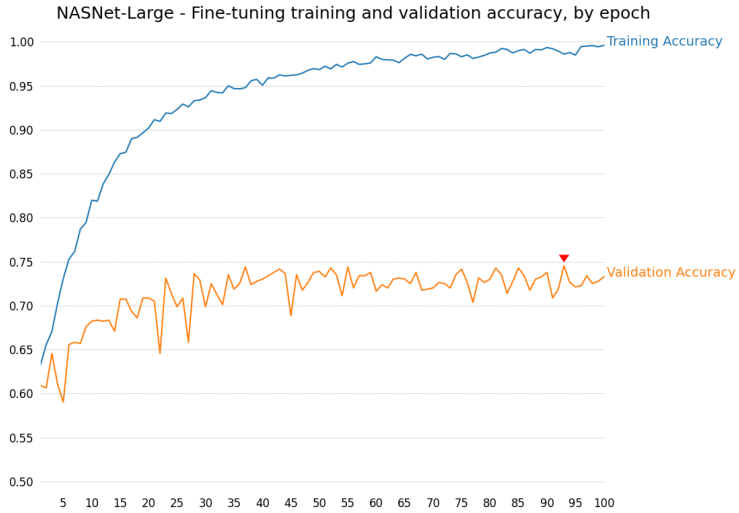

Fig. 16. NASNet-Large accuracy performance during 100 epochs of fine tuning.

Table 4. Fin-tuning Model Performance and Parameter Overview

		Accuracy		Parameters			
	Best Epoch	Validation	Training	Trainable	Non-trainable	Total	Trainable Share
EfficientNetB0	35	89.0%	94.8%	3.1 million	0.9 million	4.1 million	77.3%
VGG16	30	84.5%	93.6%	32.8 million	7.6 million	40.4 million	81.1%
Xception	8	86.3%	99.8%	6.8 million	14.1 million	20.9 million	32.5%
ResNet50V2	46	81.5%	93.1%	4.5 million	19.1 million	23.6 million	18.9%
NASNet-Large	93	74.5%	98.6%	4.6 million	80.3 million	84.9 million	5.5%

longer shows strong signs of lack of understanding. In the transfer learning stage, the lack of understanding led to large swings in model performance. The validation accuracy peaks to 81.5% at epoch 46, an increase of 7.7%.

NASNet-Large was the model that had the worst performance—in terms of accuracy—during transfer learning, and that was still the case when using fine tuning. As can be seen in Figure 16, the validation accuracy peaks at epoch 93 with an accuracy of 74.5%. Even with an accuracy increase of 8.4%, the model still trails behind the rest, leaving a gap of 5.5% to the second-worst performing model. The NASNet-Large model was fine-tuned by unfreezing 35 additional layers, resulting in an increase of 4.6 million trainable parameters. However, it is apparent from the aforementioned figure that the model struggles tremendously to understand and classify the images correctly.

Table 4 showcases the performance of each of the five models as well as their parameter metrics. It is impressive that the EfficientNetB0 model achieves such a high accuracy with an order of magnitude in difference in terms of parameter count. Additionally, it is interesting to note that even though NASNet-Large only has 5.5% of its parameters available for training in fine tuning, it still manages to overfit without reaching high levels of validation accuracy.

4.4 Performance Discussion

Seeing as the fine-tuned models performed better than their transfer learning counterparts, the fine-tuned models were selected for the final model evaluation and validation. They were evaluated against data that they had never seen before—data that had never been used for either training or

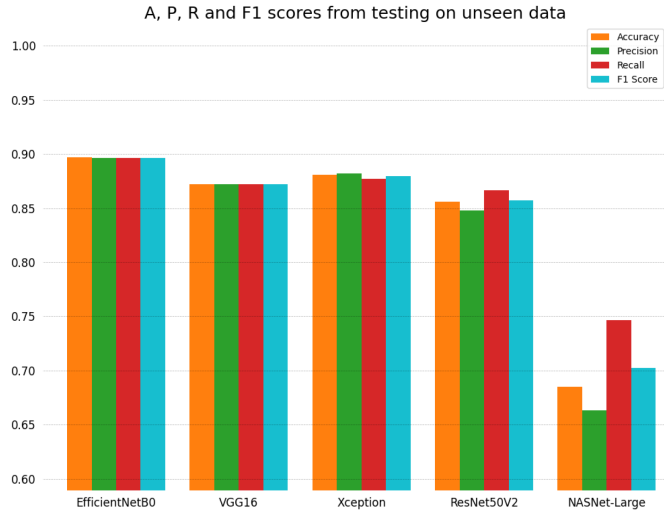

Fig. 17. Model performance in predicting unseen testing data.

Table 5. Model Performance Results from Testing on Unseen Testing Data

	Accuracy	Precision	Recall	F1 score
EfficientNetB0	0.897	0.896	0.896	0.896
VGG16	0.872	0.872	0.872	0.872
Xception	0.881	0.882	0.877	0.880
ResNet50V2	0.856	0.848	0.866	0.857
NASNet-Large	0.685	0.663	0.747	0.703

validation. The results of this evaluation were used to assess the models' generalization power as well as their performance.

As shown in Figure 17 and Table 5, NASNet-Large is still an outlier as its performance is severely behind the other four models on all metrics. EfficientNetB0 also stands out, as the best performing model with an accuracy of 89.7% on the testing data reaching near 0.9 on all metrics, a very impressive score. Its high precision shows that it has an exceptionally low false positive rate, which is essential for classifying tampered images. The high recall score shows that the model finds and classifies the majority of the tampered images. A high F1-score is indicative of a model that is uniformly good at predicting both classes correctly. Here, Table 5 shows the data represented in Figure 17.

5 DISCUSSION

In this section, we would like to shed some light on our struggle and challenges in regard to data acquisition.

(1) The problem was double faceted: there was a severe lack of quality datasets with tampered images and, for the datasets not freely available, the researchers handling the datasets appeared entirely unreachable. Most of the datasets mentioned in existing literature were not accessible. In the few cases, the obtained datasets were either of limited scope or questionable quality. This proved to be quite a challenge as well as the construction of our dataset: "Omega". As earlier

mentioned, the Omega dataset is composed entirely of freely available datasets of medium to high quality images and an even distribution of pristine and tampered images.

(2) Fine tuning increased the performance of all tested models, and some models saw huge improvements when implementing it. This could be due to the fact that the classification problem this work attempts to solve is quite different in nature from the classification in ImageNet. It is not a particular element our classification is trying to find represented in images, rather it is whether or not something within the image does not fit in. In other words, some knowledge that the models inherited from the original ImageNet weights is less useful than other knowledge. The top few layers of the models hold the most weights with most impact, as they are the most specific to the data. Therefore, changing those particular weights, as was done in this work through fine tuning, showed tremendous improvements in accuracy.

(3) As with every machine learning task, overfitting is a real concern. There is no real measurement or specification to determine precisely when a model is overfitting on data and becoming less efficient. Multiple generalizations are applicable such as when the slope of the training curve is nearing a horizontal state or when training accuracy keeps increasing without the validation accuracy increasing.

(3.a) Overfitting is very relevant for this work, especially because we are working with large-scale models and pre-trained weights. Models of this caliber exhort us to be cautious and to implement countermeasures to avoid overfitting, such as dropout. Even when only fine tuning on the last layers of the model, overfitting remains a concern. Another example of measures to prevent overfitting is: the ResNet50V2 model's feature of skipping blocks does not provide any improvements, thus also eliminating some potential overfitting. Even with counter measures and model's features directed toward battling overfitting data, the desired outcome is not always achieved.

(3.b) Overfitting was a deciding factor when choosing to opt for the EfficientNetB0 model over some of its larger, more complex, and more capable siblings. A model as powerful as EfficientNetB7 is at a much higher risk of overfitting. Had we chosen the EfficientNetB7 model, we would have had to take even more drastic measures to mitigate overfitting.

(3.c) One such measure would be increasing the size of the dataset by several magnitudes. Unfortunately, that solution was simply not feasible due to the aforementioned complications with acquiring data. That being said, with the performance seen from the EfficientNetB0, one could only expect that, given enough additional data and other measures to prevent overfitting, the EfficientNetB7 would be capable of expanding upon the performance seen in this article.

(4) Even though our initial goal of an expansive, diverse dataset was not met, the Omega dataset is still a success. The Omega dataset is filled with a mix of images of (1) variable sizes, (2) variable tampering methods, and (3) variable levels of editing quality—all of which are essential and distinguishing facets. Even with those facets, the Omega dataset still maintains an even distribution of tampered and pristine images, mitigating concerns of bias that have plagued most related works.

6 CONCLUSION

In this work, our ultimate goal was to assess the viability of treating image forgery detection as a standard image classification problem and solving it with standard image classification tools. Transfer learning using pre-trained ImageNet weights proved to be a viable option to achieve decent models for image forgery detection. In addition, fine tuning the transfer learning models to better fit them to the classification problem has proven highly beneficial. The EfficientNetB0 model reached a classification accuracy of 89.7% when evaluated against unseen data, an exceptional result that validates the approach put forth in this work. It also showed no bias to either image class and had excellent precision, recall, and F1 metrics—0.896 for all three.

The Omega dataset is a valuable asset when it comes to training classification models to detect image forgeries. Efforts to increase the size of this dataset would only improve the performance of the models. It would also serve the purpose of potentially allowing the use of more complex models—such as EfficientNetB7—without running into overfitting problems. The measures put in place in this work to prevent overfitting or bias were mostly successful; overfitting did occur in some cases.

Future work, apart from obtaining additional high quality forged images, could include looking at different ways—more effective ways—of fine tuning the models. In this work, the goal was to prove the validity of the approach rather than achieve the highest possible accuracy. Further improvements might include implementing a teacher network to generate labels for unlabeled data, which could be fed to a student network—a state-of-the-art technique when it comes to general image classification problems. This technique would circumvent the issues associated with the major shortage of labeled forgeries, which, in turn, would significantly improve the performance of the model.

REFERENCES

[1] Misinformation & Fake News. Retrieved from https://libguides.lib.cwu.edu/c.php?g=625394&p=4391900

[2] Latest News, Photos, Videos on Donald Trump Fake News. Retrieved from https://www.ndtv.com/topic/donald-trump-fake-news

[3] P. R. Gupta, D. Sharma, and N. Goel. "Image Forgery Detection by CNN and Pretrained VGG16 Model." *Advances in Intelligent Systems and Computing,* vol, Springer. 1411, DOI: https://doi.org/10.1007/978-981-16-6887-6_13

[4] M. Baviskar, S. Rathod, and J. Lohokare. 2022. A comparative analysis of image forgery detection techniques. In *Proceedings of the 2022 International Conference on Computing, Communication, Security and Intelligent Systems*, 1–6, Retrieved from https://ieeexplore.ieee.org/document/9885600.

[5] Z. Chen, J. Chen, and W. Meng. 2022. Threshold identity authentication signature: Impersonation prevention in social network services. *Concurrency and Computation: Practice and Experience* 34, 16 (2022), 1–9.

[6] N. Hebbar and A. Kunte. 2022. A deep learning framework with transfer learning approach for image forgery localization. In *Proceedings of the 2022 IEEE 3rd Global Conference for Advancement in Technology*, 1–6, Retrieved from https://ieeexplore.ieee.org/document/9971986

[7] R. Cristin, T. Daniya, and S. Divyatej. 2022. Image forgery detection using supervised learning algorithm. *Lecture Notes in Electrical Engineering,* vol. 907, Springer. DOI: https://doi.org/10.1007/978-981-19-4687-5_51

[8] D. Das and R. Naskar. 2022. Image splicing detection based on deep convolutional neural network and transfer learning. In *Proceedings of the 2022 IEEE 19th India Council International Conference*, 1–6, Retrieved from https://ieeexplore.ieee.org/document/10039789

[9] O. Mayer and M. C. Stamm. 2018. Accurate and efficient image forgery detection using lateral chromatic aberration. *IEEE Transactions on Information Forensics Security* 13, 7 (2018), 1762–1777.

[10] P. B. Shailaja Rani and A. Kumar. 2019. Digital image forgery detection techniques: A comprehensive review. In *Proceedings of the 2019 3rd International Conference on Electronics, Communication and Aerospace Technology*, 959–963, Retrieved from https://ieeexplore.ieee.org/document/8822064

[11] M. Lin, Q. Chen, and S. Yan. 2014. Network in network. In *Proceedings of the International Conference on Learning Representations*, 1–10.

[12] K. Li, G. J. Qi, K. A. Hua. 2018. Learning label preserving binary codes for multimedia retrieval: A general approach. *ACM Transactions on Multimedia Computing, Communications, and Applications* 14, 1 (2018), 2:1–2:23

[13] M. N. Nazli and A. Y. A. Maghari. 2017. Comparison between image forgery detection algorithms. In *Proceedings of the 2017 8th International Conference on Information Technology*, 442–445, Retrieved from https://ieeexplore.ieee.org/document/8080040

[14] C. M. Pun, X. Yuan, and X. L. Bi. 2015. Image forgery detection using adaptive oversegmentation and feature point matching. *IEEE Transactions on Information Forensics and Security* 10, 8 (2015), 1705–1716.

[15] G. Chierchia, G. Poggi, C. Sansone, and L. Verdoliva. 2014. A bayesian-MRF approach for PRNU-based image forgery detection. *IEEE Transactions on Information Forensics and Security* 9, 4 (2014), 554–567.

[16] G. Nirmalapriya, B. Maram, R. Lakshmanan, and M. Navaneethakrishnan. 2013. ASCA-squeeze net: Aquila sine cosine algorithm enabled hybrid deep learning networks for digital image forgery detection. *Computers and Securi*ty 128 (2023), 103155.

[17] H. Benhamza, A. Djeffal, and A. Cheddad. 2021. Image forgery detection review. *2021 International Conference on Information Systems and Advanced Technologies*, 1–7, Retrieved from https://ieeexplore.ieee.org/document/9678207
[18] S. Pradhan, U. Chauhan, and S. Chauhan. 2022. A review on digital image forgery detection. In *Proceedings of the 2022 2nd International Conference on Innovative Practices in Technology and Management*, 697–702, Retrieved from https://ieeexplore.ieee.org/document/9754125
[19] N. T. Pham and C. S. Park. 2023. Toward deep-learning-based methods in image forgery detection: A survey. *IEEE Access* 11 (2023), 11224–11237.
[20] S. Qian, J. Hu, Q. Fang, and C. Xu. 2021. Knowledge-aware multi-modal adaptive graph convolutional networks for fake news detection. *ACM Transactions on Multimedia Computing, Communications, and Applications* 17, 3 (2021), 98:1–98:23
[21] Y. Li and J. Zhou. 2019. Fast and effective image copy-move forgery detection via hierarchical feature point matching. *IEEE Transactions on Information Forensics and Security* 14, 5 (2019), 1307–1322.
[22] W. Li, W. Meng, and M. H. Au. 2020. Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in IoT environments. *Journal of Network and Computer Applications* 161, 102631 (2020), 1–9.
[23] W. Li, W. Meng, and L. F. Kwok. 2022. Surveying trust-based collaborative intrusion detection: State-of-the-art, challenges and future directions. *IEEE Communications Surveys and Tutorials* 24, 1 (2022), 280–305.
[24] W. Meng, W. Li, and J. Zhou. 2021. Enhancing the security of blockchain-based software defined networking through trust-based traffic fusion and filtration. *Information Fusion* 70 (2021), 60–71.
[25] T. J. Shah and M. Tariq Banday. 2020. A comparative study of block-based copy-move forgery detection techniques. In *Proceedings of the 2020 International Conference on Computer Communication and Informatics*, 1–6, Retrieved from https://ieeexplore.ieee.org/document/9104074
[26] W. Meng, W. Li, L. T. Yang, and P. Li. 2020. Enhancing challenge-based collaborative intrusion detection networks against insider attacks using blockchain. *International Journal of Information Security* 19, 3 (2020), 279-290.
[27] W. Meng, W. Li, and L. F. Kwok. 2017. Towards effective trust-based packet filtering in collaborative network environments. *IEEE Transactions on Network and Service Management* 14, 1 (2017), 233–245.
[28] R. Rani, A. Kumar, and A. Rai. 2021. A brief review on existing techniques for detecting digital image forgery. 2021. In *Proceedings of the 2021 6th International Conference on Image Information Processing*, 533–538, Retrieved from https://ieeexplore.ieee.org/document/9702688
[29] S. Ganguly, A. Ganguly, S. Mohiuddin, S. Malakar, and R. Sarkar. 2022. ViXNet: Vision transformer with xception network for deepfakes based video and image forgery detection. *Expert Systems with Applications* 210 (2022), 118423.
[30] F. Matern, C. Riess, and M. Stamminger. 2020. Gradient-based illumination description for image forgery detection. *IEEE Transactions on Information Forensics and Security* 15 (2020), 1303–1317.
[31] R. Poisel and S. Tjoa. 2011. Forensics investigations of multimedia data: A review of the state-of-the-art. In *Proceedings of the 2011 6th International Conference on IT Security Incident Management and IT Forensics*, 48–61, Retrieved from https://ieeexplore.ieee.org/document/5931112
[32] M. Ernawati, F. Ernawan, Y. Abbker, M. Fakhreldin, and P. W. Adi. 2022. Image splicing forgery approaches: A review and future direction. In *Proceedings of the 2022 6th International Conference on Informatics and Computational Sciences*, 134–139, Retrieved from https://ieeexplore.ieee.org/document/9930543
[33] R. R. K. and M. Wilscy. 2018. Pretrained convolutional neural networks as feature extractor for image splicing detection. In *Proceedings of the 2018 International Conference on Circuits and Systems in Digital Enterprise Technology*, 1–5, Retrieved from https://ieeexplore.ieee.org/document/8821242
[34] H. Wu, J. Zhou, J. Tian, J. Liu, and Y. Qiao. 2022. Robust image forgery detection against transmission over online social networks. *IEEE Transactions on Information Forensics and Security* 17 (2022), 443–456.
[35] A. Mazumdar and P. K. Bora. 2019. Deep learning-based classification of illumination maps for exposing face splicing forgeries in images. In *Proceedings of the 2019 IEEE International Conference on Image Processing*, 116–120, Retrieved from https://ieeexplore.ieee.org/document/8802969
[36] H.-Y. Choi, H.-U. Jang, D. Kim, J. Son, S.-M. Mun, S. Choi, and H.-K. Lee. 2017. Detecting composite image manipulation based on deep neural networks. In *Proceedings of the 2017 International Conference on Systems, Signals and Image Processing*, 1–5, Retrieved from https://ieeexplore.ieee.org/document/7965621
[37] V. Vinolin and M. Sucharitha. 2021. Hierarchical categorization and review of recent techniques on image forgery detection. *The Computer Journal.* 64, 11 (2021), 1692–1704.
[38] C. Yan, S. Li, and H. Li. 2023. TransU2-Net: A hybrid transformer architecture for image splicing forgery detection. *IEEE Access,* 11 (2023), 33313–33323, Retrieved from https://ieeexplore.ieee.org/document/10090941
[39] K. N. Singh, O. P. Singh, A. K. Singh, A. K. Agrawal. 2023. EiMOL: A secure medical image encryption algorithm based on optimization and the lorenz system. *ACM Transactions on Multimedia Computing, Communications, and Applications* 19, 2s (2023), 94:1–94:19

[40] W. Wei, S. Wang, X. Zhang, and Z. Tang. 2010. Estimation of image rotation angle using interpolation-related spectral signatures with application to blind detection of image forgery. *IEEE Transactions on Information Forensics and Security* 5, 3 (2010), 507–517.

[41] H. Wu, J. Zhou, J. Tian, J. Liu, and Y. Qiao. 2022. Robust image forgery detection against transmission over online social networks. Transactions on Information Forensics and Security 17, (2022) 443-456.

[42] M. T. H. Majumder and A. B. M. Alim Al Islam. 2018. A tale of a deep learning approach to image forgery detection. In *Proceedings of the 2018 5th International Conference on Networking, Systems and Security*, 1–9, Retrieved from https://ieeexplore.ieee.org/document/8631389

[43] K. R. Sharma, W. Y. Chiu, and W. Meng. 2023. Security analysis on social media networks via STRIDE model. In *Proceedings of the 19th International Conference on Networking and Services*, 28–33.

[44] M. S. Kaushik and A. B. Kandali. 2023. Convolutional neural network based digital image forensics using random forest and SVM classifier. In *Proceedings of the 2023 International Conference on Intelligent Data Communication Technologies and Internet of Things*, 860–865, Retrieved from https://ieeexplore.ieee.org/document/10053434

[45] X. Sun, W. Meng, W. Y. Chiu, and B. Lampe. 2022. TDL-IDS: Towards a transfer deep learning based intrusion detection system. In *Proceedings of the 2022 IEEE Global Communications Conference*, 2603–2608.

[46] M. Tan and Q. V. Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 6105–6114.

[47] T. Le-Tien, P. X. Hanh, N. Pham-Ng-Quynh, and D. Ho-Van. 2020. A combination of super-resolution and deep learning approaches applied to image forgery detection. In *Proceedings of the 2020 International Signal Processing, Communications and Engineering Management Conference*, 244–249, Retrieved from https://ieeexplore.ieee.org/document/9480963

[48] R. Joshi, A. Gupta, N. Kanvinde, and P. Ghonge. 2022. Forged image detection using SOTA image classification deep learning methods for image forensics with error level analysis. In *Proceedings of the 2022 13th International Conference on Computing Communication and Networking Technologies*, 1–6, Retrieved from https://ieeexplore.ieee.org/document/9984489

Received 15 May 2023; revised 12 November 2023; accepted 13 November 2023