HUMAN VS. AI IMAGE CLASSIFICATION

Riya Bharamaraddi, Ryan Bowering, Kayla Martinez, Adithya Ramji

CSSE/MA416 Deep Learning

10 November 2023

ABSTRACT

In this study, we explore and analyze the accuracy of neural networks being able to classify images as either real or generated by AI (Artificial Intelligence). We used a variety of deep learning techniques including creating small Convolutional Neural Networks (CNNs) and fine-tuning larger pre-trained models. We used the CIFAKE dataset and achieved over 98% testing accuracy. We also experimented with applying different filters on the images and training models on them to determine the impact of filtering images. This paper discusses the methods we used, the experiments we ran, and the challenges we faced during the study. Additionally, we will offer suggestions for future work in this field.

1. INTRODUCTION

Now, more than ever, there has been a rise in images generated by artificial intelligence and it has become increasingly difficult to distinguish them from real images. One application AI image generators are being used for is creating “deepfakes”, which are images that have been manipulated to misrepresent an event or person. Deepfakes have become extremely convincing too, being very hard to identify. Figure 1 shows two deepfake images.

Figure 1: Images of humans created by AI [1].

Intel is just one of the groups that has attempted to identify deepfakes with FakeCatcher, a platform for analyzing videos. FakeCatcher focuses on color changes in the face to infer blood flow and pays attention to specific facial regions. Intel’s method first applies a motion-magnification algorithm and then feeds the data to a neural network, which claims to have a 97% accuracy [2]. As more accurate techniques are developed to detect deepfake content, cyber criminals using deepfakes will have a harder time causing harm.

This is just one method that has been implemented using deep learning to detect deepfake content. We have implemented multiple classification techniques, with varying accuracies, in an attempt to solve the problem.

We trained two convolution neural networks, LeNet and LeNet2, from scratch to achieve a 94% and 95% validation accuracy, respectively. Pretrained models such as ResNet50, VGG16, EfficientNetB0, Densenet121, and ViT were fine-tuned.

Additionally, we were interested in understanding how well humans could identify AI images and wanted to be able to compare the model results to human performance. For this reason, we set up our own human performance test using a small sample of the CIFAKE dataset.

2. LITERATURE REVIEW

AI Image generation has exploded in popularity over the last year. Models have been rapidly improving, making it extremely difficult to distinguish between real and AI-generated images.

Though the generation of images has received a lot of attention, there has been some work done to improve the identification of AI-generated images. Bird and Lofti created the CIFAKE dataset by using a stable diffusion model to generate counterparts to the CIFAR-10 dataset of images [3]. This is the dataset we will focus on in our study.

Bird and Lofti also trained a small CNN to classify the images which achieved 92% accuracy. Additionally, they used Gradient Class Activation Mapping to view which parts of an image had the largest impact on their model's prediction. They found a fairly uniform distribution of importance across the real images while the important pieces of AI-generated images were generally more sparse [3].

On Kaggle, there are also several notebooks for this dataset. Piosenka fine-tuned an EfficientNetB0 model on the dataset and achieved an accuracy of 96% [4]. Iakubovskyi fine-tuned a Vision Transformer (ViT) and achieved 98% accuracy [5].

Applying filters is an important part of applying a convolutional neural network for image classification because it allows the network to extract important features that the filter targets and promotes weight sharing.

Gaussian blurring filter is a low pass filter that reduces the image's high frequency components. This technique uses the weighted mean, where neighboring pixels that are closer to the central pixel contribute the most to the average. The gaussian equation is shown in Equation 1,

$$G(x, y) = \frac{1}{2\pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad (1)$$

where x and y are the distances from the center of the kernel and σ is the standard deviation of the Gaussian kernel [6].

The median filter works by replacing each pixel's value with the median value of its neighbors. This is a non-linear technique and can preserve edges while reducing the noise in the image too [7].

Another commonly used filter is the Gray World algorithm, which is a white balancing filter. It assures that each image is gray, on average. Since the images are made of red, green, and blue channels, it results in the average of them being gray [8].

SRM filters have been used in previous work to extract data from deepfake images to classify them as real or fake. For example, when researchers achieved poor results on VGG16, they added an SRM filter too [9]. We added an SRM filter to see how it would affect our outcome.

3. PROCESS

3.1 Data Source

The CIFAKE dataset is made up of 32x32 pixel RGB images that were publicly available on Kaggle [3]. The dataset contains a total of 120,000 images, with 60,000 real images and 60,000 fake images.

The real images are from the CIFAR-10 dataset which consists of 60,000 color images in 10 classes, with 6,000 of each of the following categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The fake images were generated using Stable Diffusion version 1.4 and has the same class distribution as CIFAR-10.

These datasets are split equally so that there are 100,000 training images (50,000 per class) and 20,000 testing images (10,000 per class). We are then using this data set to train our own classification model to predict whether images are of two classes, real or AI-modified/generated. As the data is already split evenly, the two classes are balanced and have the same number of targets so there were not any issues in this regard.

3.2 Preprocessing

The original images are 32x32 pixel RGB images stored in jpg files which need to be converted into numerical data for analysis. We converted them into 32x32x3 tensors to feed into our models. We also normalized the data with a MinMaxScaler from Scikit-Learn, which scaled the pixel data to [0, 1] down from [0, 255]. To do this, we flattened the tensors into 1x3072 long arrays, normalized the data, and reshaped them back into 32x32x3 tensors. One of the baseline models we trained was a Logistic Regressor which cannot handle tensors, so for that model we did not reshape the arrays after normalizing.

Additionally, we wanted to experiment with applying different filters to the images. Since the AI-generated images were created using a diffusion model, they started off as complete noise. We wanted to compare a base model to one trained on images with noise filtered out, or just the extracted noise. We created

additional datasets of the images after applying a Gray World filter, Median Blur filter, Gaussian Blur filter, and SRM filter.

3.3 Manual feature extraction or engineering

We primarily used large pre-trained neural networks for this study, so we did not do any manual feature extraction. Neural networks learn their own features by design, so we start with models that have been trained on massive datasets like ImageNet and have learned their own general features of images, then train a small classifier on top of that.

3.4 Classifiers/regressors and tuning

We started with a Logistic Regressor from Scikit-Learn as a quick baseline classifier. The algorithm works to predict the probability that an image belongs to a specific class or not. It takes the output of a linear regression function as an input and uses a softmax function to estimate the probabilities of being in each class. The logistic regressor was able to achieve 70.09% training accuracy and a 68.82% test accuracy.

One of the first models we trained was VGG16, a CNN with 16 layers designed to only use 3x3 filters with a stride of 1 for the convolution layers [10]. This model was trained on ImageNet, so we loaded those weights to start with and replaced the classification head to put images in one of two classes, real or fake. We achieved an accuracy of 87.66% with this model.

Another CNN we fine-tuned was EfficientNet. It is built upon a concept called compound scaling, which refers to systematically scaling the model's dimensions (width, depth, and resolution) to increase efficiency without compromising on accuracy [11]. It provides several variants with different scaling coefficients, where each demonstrates the trade-off between model size and accuracy. We worked with EfficientNetB0, the base model, since it was the least computationally intensive variant and was sufficient for the scope of our task. The core of the model consists of a mobile inverted bottleneck convolution layer along with squeeze and excitation layers. After experimentation with different hyperparameter values, we settled on a batch size of 32 and 10 epochs. This model achieved a validation accuracy of 96.82% and a test accuracy of 96.59%. We trained another EfficientNetB0 model on augmented data that underwent rotation, width and height shifts, and horizontal and vertical shifts. We were able to achieve only around 50% accuracy in this case for the validation and test sets.

Densenet121 reduces the number of required parameters and strengthens feature propagation. It is also more accurate and efficient with shorter connections between input and output layers. DenseNet layers are narrow and each have direct access to the gradient from the loss function and the original input image. DenseNets also concatenate the output feature maps of the layer with the incoming feature maps [12]. The model was trained with a batch size of 32 and 10 epochs as well, and a train-validation split of 80/20. The model achieved a validation accuracy of 94.30% and a test accuracy of 94.07%.

ResNet50 aims to fix decreasing accuracy for when a network starts to converge. Microsoft introduced the ResNet framework in which the shortcut connections are made by skipping one or more layers to perform identity mapping. Every layer has its own weight to learn. The outputs are then added to the outputs of the stacked layers. This is done instead of directly fitting the stacked layers to the desired underlying mapping [13]. This model was also trained with a test-validation split of 80/20 and 10 epochs, and achieved a validation accuracy of 87.80% and a test accuracy of 87.97%.

We also fine-tuned a ViT model that was pre-trained on ImageNet. ViT models are Transformers adapted to do computer vision tasks instead of processing sequences of text vectors. They do this by splitting images into 16x16 chunks, then flattening those chunks into vectors to feed them into the Transformer for classification [14]. These models are some of the top performing on ImageNet, and it was also our top performing model scoring 98.65% accuracy on CIFAKE.

Although pre-trained models are known to perform better on certain classification tasks, we decided to experiment with basic CNNs created from scratch as a means for comparison to the baselines and the fine-tuned models. We built LeNet, one of the earliest convolutional networks, from scratch, and trained it on the dataset with 10 epochs and a batch size of 100. This was able to achieve a validation accuracy of 93.96%. We also created a similar model to LeNet

that we called LeNet2, where we duplicated both of its convolutional layers. This change boosted its validation accuracy to 95.20%. Additionally, we tried making a small CNN similar to LeNet but without pooling layers since modern trends are indicating a shift towards performing less pooling in order to preserve the data. This achieved 94.72% validation accuracy.

One experiment we tried was training a large pre-trained model from scratch. Rather than loading the weights for EfficientNet from training on ImageNet, we started with randomized weights and attempted to train it only on our dataset. We were not able to get above 50% validation accuracy though, which is not better than our baseline. This is likely because EfficientNet is a very large model designed to train on massive datasets like ImageNet which has over 1.2 million images, so it was not able to learn enough from CIFAKE which only has 120,000 images.

Another experiment we aimed to perform with basic CNNs was to determine the effect of changing certain hyperparameters on the resulting model performance. We varied the number of convolutional layers, the number of dense layers, and the number of neurons used by the dense layers. Using the best model from the above experiment, we decided to test the effect of regularization and dropout on the CNN. Regularization helps prevent overfitting by adding a penalty term to the loss function that discourages large weights. Dropout is a technique that randomly deactivates a fraction of neurons during training to prevent overfitting.

4. EXPERIMENTAL SETUP AND RESULTS

To compare our models to human performance, a human baseline was formed consisting of 15 randomly selected images from the dataset. The questions were placed in a survey for subjects to take at any time. Each question consisted of a 32x32 pixel image and the test subject was asked to identify if the image was human or AI generated.

A total of 63 test subjects took the human baseline test, resulting in an average score of 8.46/15 points or 56.4% and a median of 8/15 points, as shown in Figure 2.

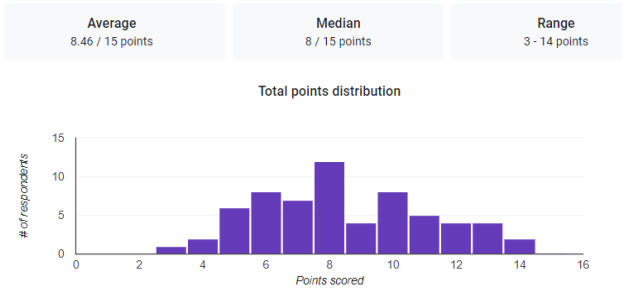

Figure 2: Results of our human baseline test distribution. The average score was 8.46/15.

We fine-tuned many pre-trained models including ResNet50, VGG16, EfficientNetB0, Densenet121, and ViT. The results from these models are shown in Table 1. The data was trained for 10 epochs with an 80/20 train-validation split.

Table 1: Results for pretrained models.

	Train accuracy (%)	**Validation accuracy (%)**	**Test accuracy (%)**
ResNet50	88.62	87.80	87.97
VGG16	87.71	88.31	87.79
EfficientNetB0	98.77	96.82	96.59
Densenet121	96.80	94.30	94.07
ViT	99.33	99.28	98.65

The Confusion Matrix for the best performing model, ViT, is shown in Figure 3.

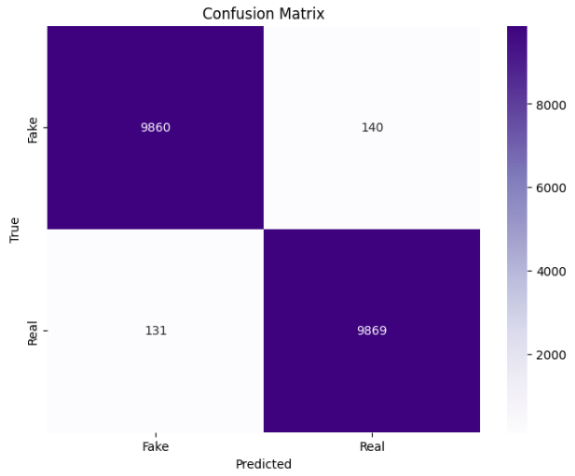

Figure 3: Confusion matrix for predicting the test set with ViT.

	precision	recall	f1-score	support
0	0.99	0.99	0.99	10000
1	0.99	0.99	0.99	10000
accuracy			0.99	20000
macro avg	0.99	0.99	0.99	20000
weighted avg	0.99	0.99	0.99	20000

Figure 4: Classification report for predicting the test set with ViT.

Many Deep Learning pipelines will use data augmentation to generate a theoretically infinite dataset from a training dataset of images by applying small random transformations to them that create new images while still maintaining their original content. We tried training EfficientNetB0 on an augmented version of our training set, but were only able to achieve 50% testing accuracy.

We also experimented with applying different filters to the images, including a Gray World filter, a Gaussian Blur filter, a Median Blur filter, and an SRM filter. A comparison of images with the different filters applied are shown in Table 2.

Table 2: Images used in experiment with different filters applied.

	Fake	Real
Original		
Gray World		
Gaussian Blur		
Median Blur		
SRM Filter		

We then trained EfficientNetB0 on the filtered dataset, since it was the model with the highest accuracy other than ViT, but was significantly faster to train than ViT. The results are shown in Table 3. The data was trained using the same number of epochs and the same train-validation split as mentioned earlier. We used LeNet for the SRM filtered images because it reduced the images to only 1 channel from 3, so we could not use any of the ImageNet pre-training for EfficientNet.

Table 3: Results for training EfficientNetB0 with filtered images.

Filter	Train accuracy (%)	Validation accuracy (%)	Test Accuracy on filtered data (%)
Gray World	96.33	51.56	50.00
Gaussian Blur	97.36	95.46	95.31
Median Blur	95.56	93.01	92.88
SRM Filter (Trained with LeNet)	92.63	85.89	85.47

We also tried using the model trained on filtered data to predict the original test data. The results of this are shown in Table 4.

Table 4: Results of training EfficientNetB0 on the filtered images and predicting the unfiltered test images.

Filter	Test Accuracy on Original Data (%)
Gray World	49.98
Gaussian Blur	50.09
Median Blur	50.12

While experimenting with the number of layers and the number of neurons used in the dense layers of the basic CNNs, we found that the worst test accuracy (<81%) was in the case where we had 2 convolutional layers, 2 dense layers, and 512 neurons per dense layer. The best test accuracy (>89%) was in the case where we had 2 convolutional layers, 4 dense layers, and 512 neurons per dense layer. Figure 5 shows a plot of how the number of convolution layers, the number of dense layers, and the number of neurons per dense layer affects the validation and test accuracy for a few selected values of the hyperparameters.

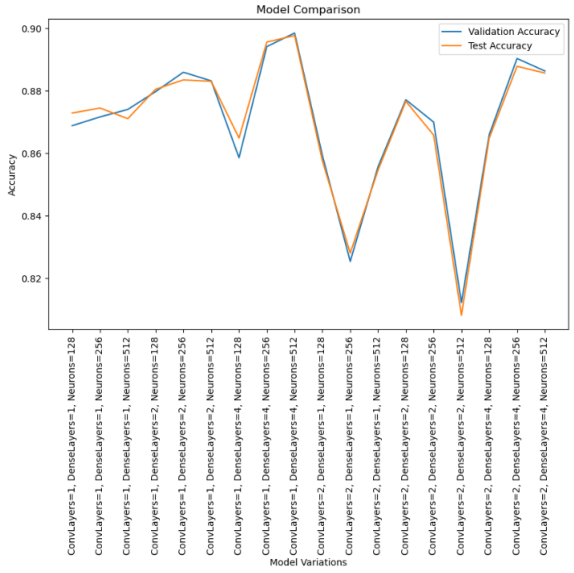

Figure 5: Effect of the number of layers and the number of neurons per dense layer on accuracy of basic CNN.

Figure 6 shows the results of varying the L2 regularization strength for the best model from the above trial, while Figure 7 shows the results of varying the dropout rate on the same model. Going from a regularization strength of 0.001 to 0.1 sharply dropped the accuracy. While dropout did not have such a drastic impact, a dropout rate of 0.2 kept the test accuracy above the 89% mark.

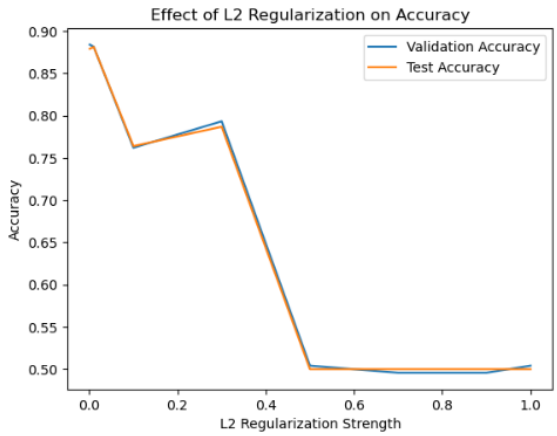

Figure 6: Effect of L2 regularization strength on accuracy of basic CNN.

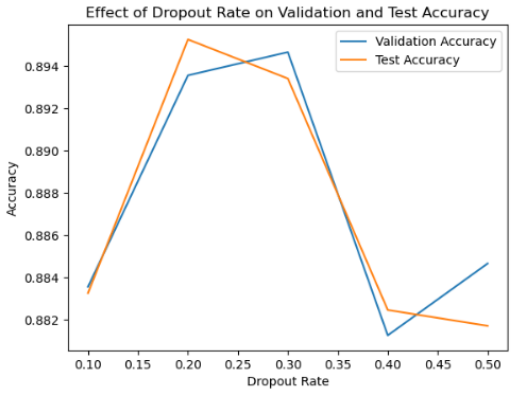

Figure 7: Effect of dropout rate on accuracy of a basic CNN.

5. DISCUSSION

Overall, our models performed very well on the original dataset. Since there are 60,000 real and 60,000 fake images, the baseline accuracy is 50%, which is what we would expect to score if we simply guessed the majority class every time. The results of our human baseline test suggest that humans might be able to do slightly better than just guessing, scoring an average of 56.4%. Since it is very hard for humans to see 32x32 images, that number might actually be slightly higher for larger images with higher resolution.

Our base models all scored at least 87%, which supports the identification of AI-generated images is a solvable problem. ViT achieved the highest accuracy, which suggests that there might be something about the images that the Transformer architecture is able to detect with its attention mechanisms but CNNs fail to capture.

Our small CNNs also performed well, scoring over 90%. Since our dataset is fairly small, these smaller CNNs might be about the optimal size to train on our data. Surprisingly, these models outperformed some of the large pre-trained models. This might be because the pre-trained models were attempting to classify images based on their content, while we were trying to classify an image based on its creator. This could suggest that the features learned by pre-trained models are not actually very useful for this domain.

The results of our hyperparameter tuning showed that accuracy was fairly consistent over a range of hyperparameters, differing at most by 10% in some extreme cases. This means that the CNN is a relatively stable architecture in general, and playing around with the hyperparameters will only lead to small improvements if any.

Image Augmentation is a commonly used technique to artificially expand the size of image datasets to improve the training of models, since a slightly rotated image will be a new image but still have content that falls into one of the classes. We experienced a significant drop in performance when we trained on augmented images, only scoring 50%. This suggests that slightly modifying an image makes it harder to identify as real or AI-generated. This also raises the question of whether or not an augmented real image is still real and whether or not an augmented AI-generated image is still AI-generated.

EfficientNet scored relatively highly on the Gaussian Blur and Median Blur filters, which are both methods of removing the noise from the images to smooth them out. This makes them slightly blurry, but EfficientNet was still able to score above 90% on them. This might suggest that the noise is not very important in identifying AI-images since removing the noise did not terribly impact the accuracy.

Applying the Gray World filter caused EfficientNet to score only 50% on the test set, which is equivalent to just guessing. The Gray World filter attempts to balance the colors to all average to gray, so the fact that it significantly hindered the accuracy of the model suggests that it might have been using some of the differences in color to make its predictions.

We trained LeNet on the SRM filtered data since the filter changed the images to a single grayscale channel rather than 3 channels for RGB. LeNet was able to achieve 85% accuracy on this data, which suggests that maybe the noise alone actually is quite important in identifying AI-generated images, although this seems to disagree with the results of training on the Median Blur and Gaussian Blur. This might be because we used a different model for the two datasets.

6. CONCLUSIONS AND FUTURE WORK

Most of our findings show high performance accuracies with the different pre-trained models. With the pretrained models, we were able to achieve over 98% testing accuracy. Our small CNNs all performed

very well on the data, most of them generating accuracies of 90% and higher, and sometimes even outperforming the pre-trained models. This might suggest that the learned features of the pre-trained models are not helpful in classifying an image as real or AI-generated. More research should be done on training smaller models from scratch on this dataset and is work that could be completed in the future.

Different filters, including the Gray World Algorithm, the Gaussian Blur, the Median Filter, and the SRM filter were applied to the EfficientNetB0 to see how they would affect the results. While the Gaussian Blur and Median filter did not significantly impact the results, the Gray World filter caused a significant drop in performance, suggesting that something about the colors might be very important for classifying images as AI-Generated. Additionally, training LeNet on the SRM filtered data achieved very high accuracy, suggesting that the noise of an image might play an important role in its classification, which could be very important. These two features should be further explored, both for classification and generation of AI images.

Additionally, this dataset was created from CIFAR-10 which contains only 10 different classes of images, compared to ImageNet's 1000 classes. These 10 classes do not include humans or text, which are commonly generated by AI models. Creation of a larger dataset that includes more of these classes for more general training would be extremely beneficial for the improvement of AI image detectors.

Finally, more work should be done towards the interpretability of these AI image detection models. If humans were able to learn from these models and better identify AI images without their help, it would be harder for misinformation to spread through fake images. This is one of the most important things that needs to be explored as AI image generators continue to improve.

7. KEY CHALLENGES AND LESSONS LEARNED

One issue we faced was the resolution of the images. These images were scaled down to 32x32 to speed up training times, but that means that some information was lost, and it made it harder for humans who took our human baseline test. We attempted to contact the creator of the dataset to get the original 512x512 images, but we were not able to reach him.

Since we ran into issues using too many server resources in the past, and since some teams have crashed the GPU servers, we chose to primarily use Google Colab for running code. However, uploading the dataset to a session took a long time, and since Colab does not retain files, we had to re-upload the data every session. Additionally, Colab has limited system RAM and only one GPU, so we had to be careful with our memory to not cause any crashes.

We also struggled a bit with converting the data to Hugging Face Datasets for training ViT. We ended up having to train directly in Kaggle where we could add the dataset to our notebook without having to download all the images, then generating the Dataset object directly from the images instead of the compressed pixel data.

8. REFERENCES

[1] M. Tsakiris, "Deepfakes: faces created by AI now look more real than genuine photos," *The Conversation*. https://theconversation.com/deepfakes-faces-created-by-ai-now-look-more-real-than-genuine-photos-197521

[2] "Intel Introduces real-Time Deepfake Detector," Intel, Nov. 14, 2022. https://www.intel.com/content/www/us/en/newsroom/news/intel-introduces-real-time-deepfake-detector.html

[3] J. Bird, A. Lotfi, (2023). CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images. arXiv preprint arXiv:2303.14126. Available: https://arxiv.org/pdf/2303.14126.pdf

[4] J. Piosenka, (2023). Transfer Learning F1 score=96% [Jupyter Notebook]. Available: https://www.kaggle.com/code/gpiosenka/transfer-learning-f1-score-96

[5] D. Iakubovskyi, (2023). CIFAKE AI generated image detection ViT [Jupyter Notebook]. Available: https://www.kaggle.com/code/dima806/cifake-ai-generated-image-detection-vit

[6] S. Tsang, (2021). Review - DBM: Deep Blur Mapping (Blur Detection) Review — DBM: Deep

Blur Mapping (Blur Detection) | by Sik-Ho Tsang | The Startup | Medium

[7] M. Ahmad and D. Sundararajan, “A fast algorithm for two dimensional median filtering,” *IEEE Transactions on Circuits and Systems*, vol. 34, no. 11, pp. 1364–1374, Nov. 1987, doi: https://doi.org/10.1109/tcs.1987.1086059.

[8] M. Maulion, “White Balancing — An Enhancement Technique in Image Processing,” *Medium*, Feb. 01, 2021. https://mattmaulion.medium.com/white-balancing-an-enhancement-technique-in-image-processing-8dd773c69f6

[9] X. Chang, J. Wu, T. Yang and G. Feng, "DeepFake Face Image Detection based on Improved VGG Convolutional Neural Network," 2020 39th Chinese Control Conference (CCC), Shenyang, China, 2020, pp. 7252-7256, doi: 10.23919/CCC50068.2020.9189596.

[10] K. Simonyan and A. Zisserman, (2015). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.15 Available: https://arxiv.org/abs/1409.1556

[11] M. Tan and L. Quoc, “EfficientNet: rethinking Model Scaling for Convolutional Neural Networks.” Sep. 11, 2020 Available: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (arxiv.org)

[12] Pablo Ruiz, “Understanding and visualizing DenseNets,” *Medium*, Oct. 18, 2018. Available: https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a

[13] “ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks,” Jan. 23, 2019. https://neurohive.io/en/popular-networks/resnet/

[14] A. Dosovitskiy, (2021). An Image is worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv preprint arXiv:2010.11929. Available: https://arxiv.org/pdf/2010.11929.pdf