DeepFake Image Detection

Jeremy Tan, Xiyue Zhang, Adler Viton

Abstract

One of the key challenges in deepfake detection is developing models that can generalize effectively to unseen data distributions. This study evaluates the performance of baseline, pretrained, and fine-tuned deep learning models for deepfake detection on two datasets: DFDC for training, validation, and holdout testing, and FakeAVCeleb as a black-box evaluation set. Fine-tuning outperformed frozen transfer learning for ResNet-18 and NASNetLarge, with EfficientNet fine-tuned with SGD achieving the highest accuracy of 77.16% on the holdout set. However, on the black-box dataset, models experienced significant performance drops due to distributional shifts. The best generalization performance was achieved by fine-tuned NASNetLarge, which outperformed the baseline by 6%, achieving 70.32% accuracy. The effects of hyperparameter choices, particularly learning rate and optimization strategy, were also analyzed. Additionally, we find that post-hoc enhancements, such as KNN and SVM classifiers, improved performance on both datasets but were less effective than fine-tuning the models directly.

1 Introduction

Manipulation of faces in digital images has become an increasing widespread problem over the years. Bad actors initially used methods, such as naive face swapping and morphing, but those methods are easily detectable with current technological advancements. The new methods of tampering, known as deepfakes, allow for a fast, realistic way to exchange faces in a video or image. The first known deepfake video was created in 2017, and current advancements has made it such that humans cannot identify a deepfake with just the human eye [1].

Bad actors have taken advantage of this technology to spread fake news, as they are able to falsify videos and images to spread misinformation on the Internet. Figure 1 shows an example of a deepfake, where the fake face cannot be discerned from the human eye only.

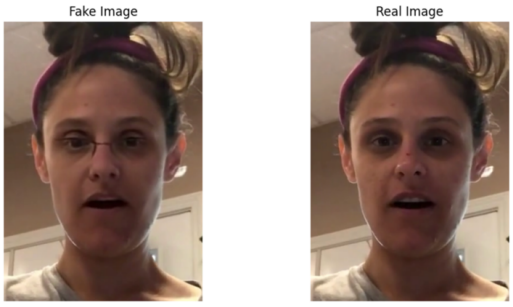

Figure 1: Side by side comparison of deepfake and real image.

Meta held a competition, called the DeepFake Detection Challenge, between 2019-2020 [2]. The purpose was to develop models that can generalize to unseen deepfakes. Thousands of participants proposed different methodologies, but no method was found to fully detect unseen deepfakes [3].

In this paper, we evaluate the performance of baseline, pretrained, and fine-tuned deep learning models across two distinct datasets: a holdout set derived from the training dataset and a completely unseen black box dataset. Our goal is to investigate how these models generalize to new data and the extent to which post-hoc enhancements, such as KNN and SVM, can improve predictive performance. We

explore a range of architectures, including ResNet, NASNetLarge, and EfficientNet, under different training configurations and optimization strategies, comparing their robustness and adaptability. By analyzing the discrepancies in model performance between the holdout and black-box datasets, we aim to highlight the challenges of generalization and the critical role of hyperparameter tuning and enhancement techniques in improving real-world applicability. This work provides actionable insights into the strengths and limitations of popular deep learning architectures and optimization strategies. The code and saved models can be found in our GitHub repository [4].

2 Related Works

The earliest work done to detect deepfakes were focused on non-deep learning approaches, analyzing JPEG compression, odd lighting of images, resampled pixels, etc. [5]. Afchar et al. proposed using CNNs, like MesoNet, to utilized dilated convolutions to encode and have broad contextual information to determine forgery [6]. Other CNNs more focused on learning how GANs performed the deepfake generation in order to spot them [7]. However, those methods don't work as well with more sophisticated deepfakes as those made by Meta [8]. Transfer learning allows models pre-trained on large-scale datasets to be fine-tuned for specific tasks, such as deepfake detection, with relatively smaller datasets [9]. This approach not only accelerates training but also enhances the model's ability to generalize across different types of deepfakes. By leveraging the knowledge captured from diverse datasets, transfer learning can help detect deepfakes that were not explicitly part of the training data, offering a more robust solution to the growing sophistication of generative techniques. Finally, instead of using the final layer of the CNN for the classification, Rafique et al. has proposed using the CNN only for deep feature extraction and using an SVM or KNN for classification [10].

3 Methodology

Figure 2 shows the overall workflow of our experiments. ResNet-18 model is used as a benchmark as it's the simplest model.

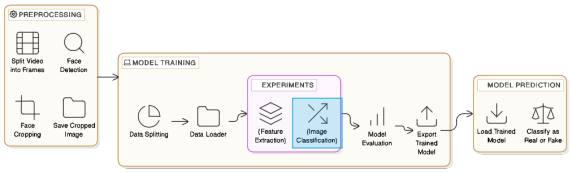

Figure 2: Workflow to run experiments.

3.1 Preprocessing and Data Augmentation

We preprocessed two datasets for our experiments: DFDC from Meta as our train, validation, and test dataset and FakeAVCeleb from Sungkyunkwan University as our blackbox dataset [2], [11]. More information about these datasets can be found in Appendix A. The DFDC dataset was chosen for its extensive collection of deep fake videos, providing a foundation for model training and evaluation within the same data distribution. The FakeAVCeleb dataset, featuring deepfakes created using diverse methods and representing varied demographics, was used to evaluate model generalization on unseen data. Frames from approximately 19,000 videos were processed using a pretrained MTCNN to detect, align, and crop facial regions, resulting in 380,000 face images across both datasets. To address class imbalance, multiple frames were sampled from real videos, while single frames were taken from fake videos. A subset of fake videos was used due to computational constraints, creating a training and validation dataset of 35,000 images, a holdout dataset of 10,000 images, and a black-box dataset of 800 images, each with a 50/50 real-fake split for class balance. The holdout dataset consisted of entirely distinct videos to prevent overfitting and ensure robust evaluation. Images were resized to 255×225 pixels, normalized, and augmented with techniques such as mirroring and color jittering during training. Validation, test, and black-box datasets remained unaltered for unbiased performance assessment.

3.2 Transfer Learning

The base model consists of ResNet-18, NASNetLarge, and EfficientNet-B0 with all layers unfrozen, trained from scratch for binary image classification. We chose EfficientNet and NASNetLarge because of their advanced architectures and better generalization capabilities compared to ResNet-18 [13], [14]. EfficientNet uses a compound scaling method to balance depth, width, and resolution. It can capture detailed and diverse features. NASNetLarge is designed through neural architecture search, optimizing its layer configurations to extract complex hierarchical patterns. See Appendix B for more information. By training the models from scratch, they are able to independently learn features relevant to the dataset.

This experiment evaluated the performance of transfer learning on DeepFake image classification. All layers for ResNet-18, NASNetLarge, and EfficientNet models were frozen to retain the general-purpose features learned from ImageNet, and the final classification layers were customized with ReLU, BatchNorm, and Dropouts to adapt to the binary classification task. The models were trained using CrossEntropyLoss, the Adam optimizer with a learning rate of 0.001, and a batch size of 64 for 50 epochs, with performance assessed on a holdout test set.

3.3 Alternate Classification Methods

This experiment intends to exploit the feature engineering capabilities of the three net-based models, ResNet, EfficientNet, and NASNetLarge, and pair them with the classification capabilities of K nearest neighbors and support vector classification as discussed in Rafique et al [10]. After creating 128-dimensional embeddings using the neural networks, KNN and SVC are compared to the traditional fully connected layer classification. A different loss function, Triplet Margin Loss, is employed in the training loop to produce advantageous embeddings for KNN and SVC. The embedding and classification models are trained on a training set, a validation set is used to find optimal hyperparameters K for KNN and C for SVC, and then an unseen holdout set is used to see results.

3.4 Tuning the Hyperparameters

Fine-tuning allowed the models to explore different hyperparameters and architectures. This experiment involved unfreezing specific layers in NASNetLarge and EfficientNet to allow task-specific learning, and implementing deeper fully connected layers with ReLU, BatchNorm, and Dropouts. We tested learning rates of 0.01, 0.1, and 0.2 with SGD and Adam optimizers to identify the best combination of parameters for each model.

3.5 Model Generalization to Black Box Dataset

The black-box dataset, composed of videos from FakeAVCeleb, differs significantly in its distribution and generation methods compared to the training dataset (DFDC) [11]. This distinction allows us to assess the model's performance on a completely unseen dataset, simulating real-world deployment scenarios. By maintaining separate, unaltered datasets for evaluation, we ensured that the model's predictions reflected genuine generalization capabilities rather than memorization of augmented patterns seen during training [15]. These steps provide a fair and reliable comparison of model performance across both familiar (holdout) and novel (black-box) data distributions.

4 Experiments

4.1 Transfer Learning

Table 1 shows the performance of the base models versus their pretrained counterparts. Although transfer learning has faster convergence speed, it's generalization without any fine-tuning didn't improve performance from the base model except for EfficentNet. In the first experiment, transfer learning was applied to enhance model performance. Pretrained ResNet-18, NASNetLarge, and EfficientNet models were used for DeepFake image classification. All layers were frozen to retain the general-purpose features from ImageNet pretraining, while the final classification layers were customized with ReLU for

non-linearity, BatchNorm for normalization, and Dropouts to mitigate overfitting. These customized layers were trained specifically to classify images as real or fake.

Model	Base Model	Pretrained Model
ResNet-18	71.66	57.95
EfficientNet	58.67	76.11
NASNetLarge	72.31	71.38

Table 1: Comparison of Base and Pretrained Models Accuracies.

Loss Function: CrossEntropyLoss; Optimizer: Adam with a learning rate of 0.001; Batch size: 64. Models were trained for 50 epochs and evaluated on a holdout test set.

ResNet-18 performs worse for transfer learning compared to the base model because its simpler architecture with only 18 layers limits its ability to capture complex patterns. When frozen, its pretrained features from ImageNet do not align well with the specific task of DeepFake detection. In contrast, EfficientNet and NASNetLarge perform better in transfer learning because of their advanced architectures. They extract richer and diverse features that can generalize better to new tasks.

However, the results from transfer learning are still not satisfactory. To further improve model performance, we explore additional techniques in the following experiments such as K-Nearest Neighbors (KNN) and Support Vector Classifier (SVC) using embeddings from the pretrained models, as well as fine-tuning specific layers to allow the models to learn task-specific features. These methods aim to address the limitations of frozen layers and enhance the models' ability to adapt to the dataset.

4.2 Experiment 2: Alternate Classification Methods

The second experiment incorporates alternative machine learning methods into the classification pipeline. Neural network embeddings are paired with K-nearest neighbors (KNN) and support vector classification (SVC), replacing the traditional fully connected layer. This approach leverages the feature extraction capabilities of ResNet-18, EfficientNet, and NasNetLarge while utilizing KNN and SVC for classification. To train models for embedding rather than classification, a metric learning loss function is required instead of traditional cross-entropy loss. Triplet Margin Loss was chosen for this purpose, as it minimizes embedding distances for images in the same class while maximizing them for images from different classes. This loss function operates by creating image triplets: an anchor image, a positive image from the same class, and a negative image from a different class. The Triplet Margin Loss formula is:

$$\mathcal{L} = \max\big(0, d(a,p) - d(a,n) + m\big)$$

where: a is the anchor, p is the positive sample, n is the negative sample, $d(\cdot,\cdot)$ represents the distance metric (Euclidean distance in this experiment), m is the margin (a positive scalar).

In practice, validation images are passed through the embedding model and then classified with the KNN or SVM trained on the training data. After training the embedding models and classifiers KNN and SVC on the training set of data, a validation accuracy is taken as a function of K for K nearest neighbors in order to find the optimal K for each embedding model. A validation accuracy as a function of C (inverse of regularization constant of SVC) is used to find the optimal hyperparameter for support vector classification. The results of this experiment are shown in the table below with Appendix C showing how we chose C and K:

Model	DL Classifier Accuracy (%)	KNN Accuracy (%)	SVC Accuracy (%)
ResNet-18	71.66	73.08	73.44
EfficientNet	58.67	75.29	75.33
NasNetLarge	72.31	72.72	72.84

Table 2: Comparison of Embedding Models Across DL Classifier, KNN, and SVC.

EfficientNet had the most significant improvement and best validation and holdout accuracy. Across all three types of embedding models, using KNN classification improved accuracy. Using SVC improved the accuracy even more on all three models, but not by a significant margin from KNN. The best-combined model was an EfficientNet embedding model paired with an SVC classification.

4.3 Experiment 3: Fine-Tuning

To improve model performance, specific layers in NASNetLarge and EfficientNet were unfrozen while retaining frozen layers for ResNet-18. The goal was to allow the models to learn task-specific features from the dataset.

For ResNet-18, all pretrained layers were kept frozen, while the fully connected layers were customized with ReLU, BatchNorm, and Dropouts to adapt to the task. In contrast, NASNetLarge and EfficientNet were fine-tuned by unfreezing specific deeper layers, allowing them to learn task-specific features from the dataset. For NASNetLarge, layers cell_17, cell_18, and cell_19 were unfrozen, while for EfficientNet, layers features.6, features.7, and features.8 were unfrozen.

Hyperparameters: learning rates 0.1, 0.01, and 0.2; Optimizers Adam and SGD; Loss Function weighted crossEntropyLoss.

Optimizer	**Learning Rate**	**ResNet-18**	**EfficientNet)**	**NASNetLarge**
Adam	0.01	64.35	64.35	66.72
Adam	0.1	64.35	50.07	49.93
Adam	0.2	53.47	53.80	49.93
SGD	0.01	73.53	77.16	69.36
SGD	0.1	70.26	72.31	70.22
SGD	0.2	66.93	69.79	68.71

Table 3: Comparison of Optimizer Accuracy Performance Across Models and Learning Rates.

The results showed that EfficientNet, fine-tuned with SGD at a learning rate of 0.01 performed best, achieving an accuracy of 77.16%. NASNetLarge, also trained with SGD and the same learning rate, achieved 73.53%. ResNet-18 showed improvement compared to Experiment 1, achieving 64.35% accuracy with Adam at a learning rate of 0.01. Interestingly, ResNet-18 outperforms NASNetLarge in certain configurations. This might be because its simpler architecture is less prone to overfitting when trained on smaller datasets. EfficientNet and NASNetLarge show significant drops in accuracy at higher learning rates (0.1 and 0.2) with Adam because of their sensitivity to aggressive weight updates that may destabilize training. While Adam converges faster initially, it can settle into suboptimal minima with smaller datasets. SGD with momentum is better at refining task-specific weights.

4.4 Experiment 4: Model Generalization to BlackBox Dataset

After training and fine-tuning all the models, evaluating model performance solely on the holdout set does not fully capture its generalization capability. To ensure the robustness of these models, we evaluated the generalization performance on the unseen dataset, FakeAVCeleb. Fig. 3 shows the top 10 models in comparison with the baseline model we used (ResNet-18). Appendix D shows the full table of the performance of all 30 models.

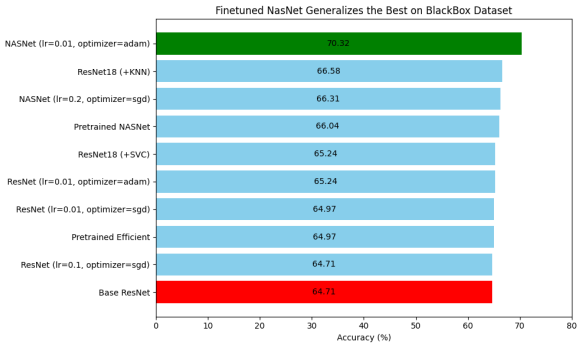

Figure 3: Top 10 Model Peformance on Blackbox dataset.

Looking at the performance of the top 10 models, the results reveal significant variations in generalization capability when transitioning from the holdout dataset to blackbox dataset. The best-performing models on the blackbox dataset generally showed lower accuracies compared to the performance on the holdout-set, showing challenges in generalization on unseen data distributions.

5 Discussion and Conclusion

On the holdout set, models generally performed better, with the baseline ResNet-18 achieving a strong accuracy of 71.66%. The highest-performing model on the holdout set was EfficientNet fine-tuned with SGD at a learning rate of 0.01, achieving an accuracy of 77.16%. This demonstrates the effectiveness of fine-tuning and optimized training strategies in improving model performance when evaluated on data with similar distributions to the training set. In contrast, all models struggled on the black-box dataset, showcasing the challenges of generalizing to unseen data distributions. Despite this, fine-tuning proved crucial for improving generalization performance, with the fine-tuned NasNetLarge model achieving 6% higher accuracy than the baseline ResNet-18.

Transfer learning proved only useful when fine-tuning for both Resnet-18 and NasNetLarge as their respective base models outperformed the frozen models. The DFDC dataset primarily consists of deepfake videos created using specific generation techniques, and fine-tuning allowed the models to adapt their feature representations to these unique artifacts. In contrast, frozen models relied on generic features from pretraining, which may not fully capture the localized inconsistencies and subtle manipulations characteristic of the DFDC deepfakes. However, EfficientNet did not follow this trend, with the frozen pretrained model outperforming its base counterpart. This discrepancy can be attributed to EfficientNet's high complexity and parameter count, which make it reliant on large and diverse datasets for effective training [14].

When fine-tuning these models, lower learning rate achieved better performance for all models. For architectures like EfficientNet and NasNetLarge, which are highly parameterized, a lower learning rate is crucial. These models have complex weight structures that are finely tuned during pretraining, and abrupt changes caused by higher learning rates can destabilize these weights, reducing their effectiveness in capturing generalizable features [13], [14]. Conversely, ResNet-18, which has a simpler architecture, also benefits from lower learning rates as they help refine its pretrained weights without overspecialization the training set [16]. Investigating learning rates lower than .01 in the future would have been a better approach for this experiment. This was especially important when we tested different optimizers, as Adam performed significantly worse than SGD on the holdout set. With the high learning rates combined with Adam, the models overfit to the train and validation set while SGD led to less overfitting. However, the adaptability of the Adam optimizer compensated for these hyperparameter sensitives, allowing it to generalize better for NasNetLarge.

SVM and KNN classifiers showed limited improvements, suggesting future efforts focus on optimizing embedding models alongside classifiers.

References

[1] P. Korshunov and S. Marcel, "Deepfake detection: humans vs. machines," *arXiv [cs.CV]*, Sep. 07, 2020. [Online]. Available: http://arxiv.org/abs/2009.03155

[2] "The DeepFake Detection Challenge Dataset," 2020. [Online]. Available: http://arxiv.org/abs/2006.07397

[3] "Deepfake Detection Challenge Results: An open initiative to advance AI." Accessed: Dec. 12, 2024. [Online]. Available: https://ai.meta.com/blog/deepfake-detection-challenge-results-an-open-initiative-to\protect\penalty-\@M-advance-ai/

[4] *ECE661–DeepFake-Detection*. Github. Accessed: Dec. 12, 2024. [Online]. Available: https://github.com/vivianzzzzz/ECE_661_Deep_Fake_Image_Detection/

[5] H. Farid, "Image forgery detection," *IEEE Signal Process. Mag.*, vol. 26, no. 2, pp. 16–25, Mar. 2009.

[6] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "MesoNet: A compact facial video forgery detection network," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, Dec. 2018, pp. 1–7.

[7] Y. Li and S. Lyu, "Exposing DeepFake videos by detecting face warping artifacts," *arXiv [cs.CV]*, Nov. 01, 2018. [Online]. Available: http://arxiv.org/abs/1811.00656

[8] S. Pashine, S. Mandiya, P. Gupta, and R. Sheikh, "Deep fake detection: Survey of facial manipulation detection solutions," *arXiv [cs.CV]*, Jun. 23, 2021. [Online]. Available: http://arxiv.org/abs/2106.12605

[9] M. Masood, M. Nawaz, A. Javed, T. Nazir, A. Mehmood, and R. Mahum, "Classification of deepfake videos using pre-trained convolutional neural networks," in *2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, IEEE, May 2021, pp. 1–6.

[10] R. Rafique, R. Gantassi, R. Amin, J. Frnda, A. Mustapha, and A. H. Alshehri, "Deep fake detection and classification using error-level analysis and deep learning," *Sci. Rep.*, vol. 13, no. 1, p. 7422, May 2023.

[11] H. Khalid, S. Tariq, M. Kim, and S. S. Woo, "FakeAVCeleb: A Novel Audio-Video Multimodal Deepfake Dataset," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=TAXFsg6ZaOl

[12] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *arXiv [cs.CV]*, Apr. 11, 2016. [Online]. Available: http://arxiv.org/abs/1604.02878

[13] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," *arXiv [cs.CV]*, Jul. 21, 2017. [Online]. Available: http://arxiv.org/abs/1707.07012

[14] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional Neural Networks," *arXiv [cs.LG]*, May 28, 2019. [Online]. Available: http://arxiv.org/abs/1905.11946

[15] A. Ballas and C. Diou, "CNN feature map augmentation for single-source Domain Generalization," *arXiv [cs.CV]*, May 26, 2023. [Online]. Available: http://arxiv.org/abs/2305.16746

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv [cs.CV]*, Dec. 10, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

Appendix A

Feature	DFDC Dataset	FakeAVCeleb Dataset
Source	Meta (Facebook)	Sungkyunkwan University
Purpose	Benchmark for deepfake detection models	Evaluation of deepfake generalization
Size	Over 100,000 videos	˜1,000 videos
Labeling	Fully labeled (real vs. fake)	Fully labeled (real vs. fake)
Deepfake Generation	Multiple generation techniques including GAN-based methods and facial reenactment	Variety of techniques, distinct from DFDC
Demographic Diversity	Includes a range of faces but limited in diversity	More diverse demographics in subjects
Dataset Composition	Split into train, validation, and test sets	Used as a black-box evaluation dataset
Video Quality	High-quality videos, varied resolutions	Varied resolutions, focusing on realism
Challenges	Intra-dataset consistency, similar patterns in real/fake	Inter-dataset distribution shift, unseen patterns
Main Use Case	Model training and evaluation on a known data distribution	Testing model generalization on unseen data

Table 4: Comparison of DFDC and FakeAVCeleb Datasets

Appendix B

Feature	ResNet-18	NASNetLarge	EfficientNet
Architecture Type	Residual Network (skip connections)	Neural Architecture Search (automated design)	Compound Scaling (width, depth, resolution)
Depth	18 layers	Variable depth (searched architecture)	Configurable (EfficientNet-B0 to B7)
Number of Parameters	11M	88M	5.3M (EfficientNet-B0)
Training Efficiency	Moderate	Computationally expensive	High efficiency due to scaling
Feature Extraction	Basic features with skip connections	Complex, hierarchical features	Efficient multi-scale feature extraction
Pretraining Availability	Widely pretrained on ImageNet	Pretrained on ImageNet and other datasets	Pretrained on ImageNet
Strengths	Simplicity, ease of training	High accuracy, automated design	High efficiency and accuracy tradeoff
Weaknesses	Limited scalability for deeper architectures	Very large and computationally intensive	Requires fine-tuning for optimal performance
Best Use Cases	General-purpose classification	Tasks requiring high accuracy and deep architectures	Scenarios with limited resources and need for high efficiency

Table 5: Comparison of ResNet-18, NASNetLarge, and EfficientNet

Appendix C

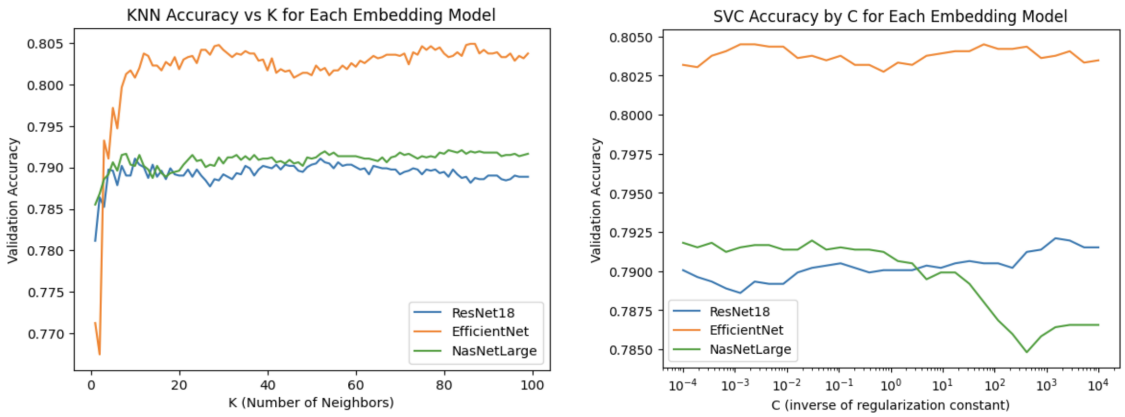

Figure 4: Accuracy on Validation Set with varying hyperparameters for KNN and SVC classifiers.

Embedding Model	**Optimal K**	**Optimal C**
ResNet-18	10	1487
EfficientNet	86	1.269e-3
NasNetLarge	81	3.039e-2

Table 6: Optimal K and C Values for Embedding Models.

Appendix D

Model	**Accuracy (%)**	**Weighted Precision**
Base ResNet	64.71	0.6746
Base NASNet	56.95	0.7615
Base Efficient	55.61	0.7575
Pretrained ResNet	58.29	0.6399
Pretrained NASNet	66.04	0.6767
Pretrained Efficient	64.97	0.6714
ResNet (lr=0.01, optimizer=adam)	65.24	0.6692
NASNet (lr=0.01, optimizer=adam)	70.32	0.7280
Efficient (lr=0.01, optimizer=adam)	55.61	0.7575
ResNet (lr=0.1, optimizer=adam)	44.65	0.4291
NASNet (lr=0.1, optimizer=adam)	53.48	0.2860
Efficient (lr=0.1, optimizer=adam)	46.52	0.2164
ResNet (lr=0.2, optimizer=adam)	49.20	0.5593
NASNet (lr=0.2, optimizer=adam)	53.48	0.2860
Efficient (lr=0.2, optimizer=adam)	53.21	0.5270
ResNet (lr=0.01, optimizer=sgd)	64.97	0.6487
NASNet (lr=0.01, optimizer=sgd)	63.37	0.6342
Efficient (lr=0.01, optimizer=sgd)	57.75	0.7011
ResNet (lr=0.1, optimizer=sgd)	64.71	0.6462
NASNet (lr=0.1, optimizer=sgd)	63.90	0.6389
Efficient (lr=0.1, optimizer=sgd)	62.83	0.6817
ResNet (lr=0.2, optimizer=sgd)	64.17	0.6454
NASNet (lr=0.2, optimizer=sgd)	66.31	0.6623
Efficient (lr=0.2, optimizer=sgd)	59.63	0.6108
ResNet18 (+KNN)	66.58	0.6705
ResNet18 (+SVM)	65.24	0.6546
EfficientNet (+KNN)	62.30	0.7185
EfficientNet (+SVM)	61.76	0.7137
NasNetLarge (+KNN)	64.17	0.6434
NasNetLarge (+SVM)	64.71	0.6472

Table 7: Model Performance on BlackBox Dataset