Deep neural networks for facial recognition.

Woo Jin Lee
Department of Electrical, Computer and Software Engineering
The University of Auckland
Auckland, New Zealand

***Abstract* - This review examines several deep learning architectures for facial recognition, focusing on VGG, ResNet, Inception v1, v4, and EfficientNet. EfficientNet-B0 outperforms VGG-16 and matches ResNet-152 in accuracy while being more computationally efficient when using the LFW dataset for facial classfication. The qualitative and quantitative evaluations carried out finds EfficientNet-B0 particularly interesting for project 2, based on its superior performance and lower computational demands in facial classification tasks.**

I. Literature Review

In the last decade, facial recognition approaches have been increasingly utilising deep neural networks to improve performance [1], especially in the recognition phase [2]. This literature review focuses primarily on four network architectures specified in the assignment brief: VGG (Visual Geometry Group network) [3], ResNet [4], GoogLeNet (Inception v1) [5], and Inception v4 [6] through the lens of facial recognition where relevant. Additionally, a review of EfficientNet [7] is also carried out.

A. VGG

VGG is an architecture developed by Simonyan and Zisserman of the Visual Geometry Group at the University of Oxford, from which it derives its name. The architecture follows closely to a conventional convolutional feed-forward network, with the key difference being its high depth for its time, with the deepest network possessing 19 weight layers.

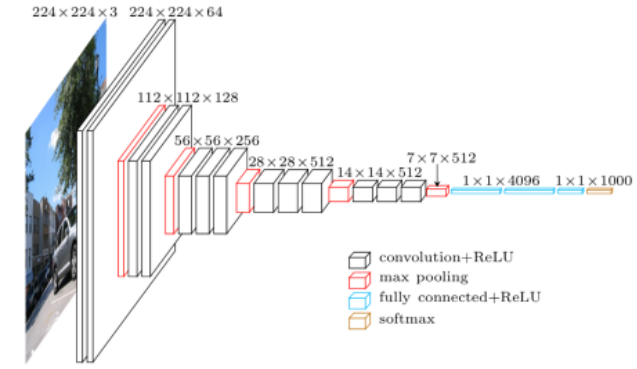

Figure 1: An illustration of the VGG-16 network architecture [8].

As seen in Figure 1, layer configurations remain consistent throughout the layers, with 3x3 convolutional layers and ReLU activations being complemented with 2x2 max pooling layers spaced out in between. Fully connected dense layers follow this, culminating in a SoftMax layer for classification.

The 3x3 kernel was specifically chosen to minimise the receptive field of the convolutional layers without losing the notion of directionality (left, right, up, down, centre). This differs from many of its contemporaries, which used larger receptive fields [9], [10], [11] near the input layer. We can recognise an effective receptive field of 5x5 or 7x7 on further analysis as the 3x3 layers with stride 1 are stacked in clusters of 2 or 3, respectively. This results in fewer total parameters to train while potentially increasing discriminatory potential than a single 5x5 or a 7x7 convolutional layer as more non-linear activation layers are applied.

VGG further demonstrated the importance of depth for convolutional feature extraction through its excellent performance at the 2014 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). With an ensemble of 7 VGG models, it placed 2nd for the classification task – second only to GoogLeNet (but still being competitive with VGG having only a 0.7% higher top-5 error rate[1] than GoogLeNet in single-net benchmarks) [12].

However, due to its high depth, the VGG architecture suffers from a high parameter count, with ~144 million parameters for VGG-19. Consequently, both training and inferencing come at a high computational cost (with VGG-19 requiring ~19.6 billion floating-point-operations (FLOPs) for a single forward-pass [4]) and results in large model sizes, limiting its usage in real-time facial recognition applications.

B. ResNet

With conventional VGG-style networks, we generally observe lower error in networks with deeper depths of tens of layers. However, as depth increases further, we observe a saturation, then an increase in error [4]. Notably, this phenomenon cannot be attributed to overfitting as the training error increases alongside the test error [4]. The ResNet architecture aims to directly address this *degradation* phenomenon.

Consider a shallow model. It follows that a deeper model should achieve analogous or greater accuracy than the shallow model as, at minimum, the additional layers in the deeper model can learn the identity function. However, experiments showed this to be false [4], with deeper models struggling to learn the identity function and failing to obtain comparable performance. By contrast, learning the zero function is easier. Random initialisation of weights around the mean of zero and L1 and L2 regularisation are well-known techniques that bias the weights towards zero [13].

The creators of the residual network first reformulates the problem of transforming the input to the output (as done in conventional feedforward networks) to a problem of finding the difference between the input and the output – the *residual*. With this formulation, when the learned function tends towards the zero function, the total model function tends towards the identity function (as the learned function computes the **difference** between the input and the output).

[1] The percentage of times the true label was not included in the top 5 highest probability predictions.

To implement this formulation, the residual network introduces skip connections, which directly add the output from a previous layer (or the input layer) to a deeper layer, bypassing the intermediary weight layers. This is illustrated in Figure 2.

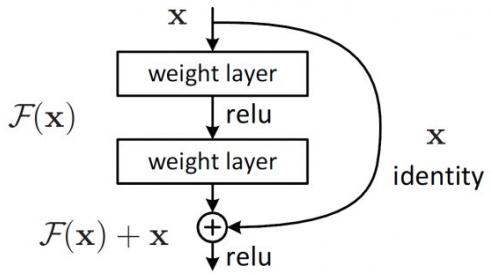

Figure 2: A diagram of a single residual block.

The network can now be viewed as a series of *residual blocks*, where the swath of a skip connection informs block boundaries. The addition result is put through a ReLU activation to introduce non-linearity. After a series of residual blocks, the output is put through dense, fully connected layers, and a final SoftMax is used for classification.

It is important to note that the dimensions of the identity and residuals must match for an element-wise addition operation. When pooling (or stride-2 convolutions) changes the dimensionality of the residual, 1x1 convolutions with stride-2 within the skip-connections to match the dimensionality of the identity were found to produce optimal performance [4].

The result is the ability to train **much** deeper models, with hundred-layer models being trainable without encountering the degradation problem. This remained true even with a 1202-layer model, although the model did achieve worse performance than the 110-layer model due to overfitting [4]. The increased depth also corresponded to better performance as an ensemble of six ResNet models with different depths placed first place in the ILSVRC 2015 competition with a top-5 error rate of 3.57% [14], which is almost half the error rate of VGG and GoogLeNet from ILSVRC 2014 (7.32% and 6.66% respectively) [12]. This performance also led to further notable successes in facial recognition tasks, such as with the ArcFace model, which utilises ResNet architecture [15].

Notably, this performance was not achieved at the cost of exponential computational complexity, with ResNet-152 having ~12 billion FLOPs [16]. This is still less than VGG's 144 million parameters, resulting in 19.6 billion FLOPs per forward pass [4]. However, due to its depth, it inevitably still has relatively high computational and memory requirements (~60 million parameters [7]) compared to shallower models such as GoogLeNet.

C. GoogLeNet (Inception v1)

Although perhaps oversimplified, GoogLeNet can be thought of as a chain of inception modules. The inception module aims to capture multi-scale information (i.e., finer **and** coarser details) while being more computationally efficient than simply adding depth (as was done in VGG).

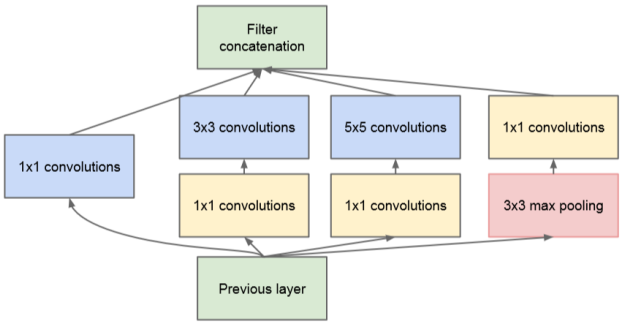

Figure 3: A diagram of the inception module.

As seen in Figure 3, a single inception module contains multiple convolution kernel sizes – namely 1x1, 3x3 and 5x5 which help capture finer details as well as coarser representations. Unlike previous architectures we examined, such as VGG and ResNet, these convolutions are not done sequentially. Instead, the inception module performs them in parallel, growing the model horizontally. The result of these convolutions, alongside the result of a 3x3 pooling layer, is concatenated channel-wise and forms the output of a single inception module.

The 1x1 convolutional layers that prepend 3x3 and 5x5 convolutions – sometimes referred to as the *bottleneck layer* [4] – serve to increase representational power (with additional activation layers) as well as reduce the dimensionality of the data without incurring significant performance penalties [5]. This principle is also used in deeper ResNets (for architectures with 50+ layers) [4].

The 1x1 convolutional layer after the 3x3 max pooling is motivated by the same dimensionality matching problem encountered by ResNet's skip connection layers post-pooling.

Its architecture is not as simple as VGG. Still, it has superior performance and lower computational requirements (~2 billion FLOPs per forward pass [17]), making it suitable for a wider problem space (e.g. facial recognition in resource-constrained environments). However, the relatively shallow nature of the network does constrain its prediction accuracy compared to deeper models such as ResNet-152, as discussed before.

D. Inception v4

The inception v4 was introduced alongside Inception-ResNet-v2, which shares some elements of the 'pure' inception v4 model (such as the stem) alongside ideas of residual connections borrowed from ResNet. It primarily aims to match parameter count and computational complexity to the pure inception v4 model. As the brief explicitly states, "Inception v4", this literature review will focus solely on the *pure* inception v4 architecture.

Inception v4, as its name implies, follows closely to the objectives of the original inception architecture while trying to improve performance. It aims to capture multi-scale details while balancing computational complexity.

To this end, several changes were made. First, specialised inception modules (A, B and C) can be seen at different depths of the architecture, as shown in Figure 4.

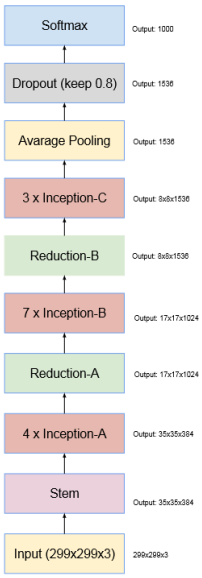

Figure 4: A high-level architecture diagram of Inception v4.

Inception module A uses dense 1x1 and 3x3 convolution filters, whereas inception module B uses coarser 7x7 convolution filters, which have been factorised to 7x1 and 1x7 filters. A similar factorised filter can be seen in module C but with 3x1 and 1x3 filters.

Factorised filters where a 1xN then Nx1 filter may be used in sequence to have an equivalent receptive field of an NxN dense filter is another notable change from v1.

The model makes further improvements to previous models, improving top-5 error rates by 0.7% over ResNet-151 on single model benchmarks. However, due to model complexity, it does require a relatively sizable 12 billion FLOPs for a single forward pass – comparable to the ResNet. This makes it equally infeasible for face recognition in resource-constrained environments. It should be noted that thanks to the various tricks covered previously, the model size is ~25% less than ResNet-152 at 48 million parameters compared to ResNet's 60 million [7].

E. EfficientNet

EfficientNet is not a singular model. Instead, it is a baseline model alongside a principled compound scaling approach that aims to address the challenge of scaling neural networks to improve prediction accuracy while remaining as computationally efficient.

Architectures, such as ResNet and VGG, demonstrate that deeper networks typically yield lower error rates; however, they are accompanied by increased computational cost and memory requirements, as we have discussed. Scaling up networks by merely increasing depth, width, or input resolution has been a common approach, but this often leads to suboptimal results and significant computational overhead. EfficientNet introduces a more principled and systematic way to scale networks through a compound scaling method, balancing depth, width and input resolution simultaneously to achieve state-of-the-art performance across various computer vision tasks [7].

The core idea behind EfficientNet is to scale up a baseline model (named EfficientNet-B0) developed via a multi-objective neural architecture search that takes both accuracy and computational complexity as measured by FLOPs. This is then scaled using the compound scaling method [7].

The results show that the EfficientNet models can yield **comparable or better accuracies than all four models analysed thus far** while remaining much more computationally efficient. For example, EfficientNet-B2 achieves 5.1% top-5 error rates, comparable with Inception-v4's 5%. Although it technically loses by 0.1%, EfficientNet-B2 achieves this with only 9.2 million parameters and 1 billion FLOPs compared to Inception-v4's 48 million parameters and 13 billion FLOPs [7].

By virtue of its performance and the principled scaling idea alone, this model is worthy of comparison to the baseline models. However, the relative computational efficiency of the model could be of particular interest to several common applications of facial recognition, such as those on resource-constrained systems (such as the NVIDIA Jetson Nano used in project 2).

II. DATASET REVIEW

A. Labelled Faces in the Wild (LFW)

Originally released in 2007, LFW is a dataset comprised of 13233 distinct 250x250 images of 5749 individuals originally designed for face verification [18]. The dataset contains faces "in the wild". That is, images of faces in uncontrolled real-world environments, where the face may be partially occluded (sunglasses, masks, gates, etc) and depict differing emotions. Furthermore, backgrounds may be messy or even contain other partial faces. Notably, this dataset is unbalanced in various aspects. The first and most apparent is in the samples per class, with only 1680 classes having two or more images and ~10x sample count difference between the most numerous (George W Bush) and 10th most numerous (John Ashcroft). Beyond just class imbalance, age, ethnicity, lighting and pose were also not considered during the creation of the dataset [18] (since the original release, a new 'aligned' dataset has also been released which aims to align facial pose), which limit its use in assessing performance in those areas.

B. Flickr-Faces-HQ Dataset (FFHQ)

FFHQ is a high-resolution face dataset originally designed to support the training of generative models. As such, no labels are placed on the images, making them unsuitable for direct use in facial classification tasks. The dataset consists of 70000 distinct images crawled from Flickr – a public image-sharing site, and thus inherits the biases seen on it. An example of this could be low-light scenarios where users of Flickr would be unlikely to share what they consider a 'bad photograph' due to the lighting condition. FFHQ provides three main views: 1) a full, uncropped image from Flickr, 2) aligned and cropped images of faces at 1024x1024 and 3) aligned and cropped images of faces at 128x128.

Automatic filters were used for alignment and cropping, and a workforce of manual human labourers was used to remove edge cases such as statues, paintings or photos of photos from the data [19]. Its high resolution makes it exceptionally well suited for capturing facial skin textures and jewellery compared to other, lower-resolution models. Unlike LFW, FFHQ also explicitly aims to provide a diverse range of ages, ethnicities, viewpoints, lighting, accessories, and image backgrounds [20].

C. Tufts Face Dataset

Tufts Face dataset is unique in that it focuses on multi-modal face analysis. It includes more than 10000 images in various modalities of face data, such as images from driver's licenses and passwords, infrared images, sketches, and even 3D models. The limitation of this dataset stems primarily from its comparatively limited individual count (113) and controlled collection environment. The collection environment naturally differs from LFW and FFHQ as they had to collect various modalities of the same person. This constraint to a laboratory collection environment may limit its application in messier real-world scenarios. Furthermore, although the individuals were of nationalities, gender identities, ages, and ethnicities, they all attended Tufts University, inheriting all biases from the entry & selection processes [21].

D. UTKFace

UTKFace is a facial image dataset containing over 20,000 images labelled with age, gender, and ethnicity, with optional landmarking making it useful for tasks like age estimation, gender classification, and ethnicity recognition. The dataset spans a wide age range (0 to 116 years) and includes images with diverse real-world conditions, such as varying lighting, facial expressions, and slight occlusions [22]. Similar to LFW, UTKFace has a relatively low image resolution at 200x200 pixels when compared to FFHQ, which limits its use in tasks requiring fine-grained detail, such as skin texture analysis. UTKFace is also more structured compared to LFW and FFHQ, with less background noise and better-cropped faces. Despite this, it suffers from demographic imbalances, especially with underrepresented ethnicities [23].

III. METHODOLOGY

This quantitative examination aims to get a broad-scoped understanding of the architectural behaviours. That is, instead of simply selecting the best-performing models, we select distinct performance profiles, which we can then infer from. Based on the qualitative literature review, VGG (16-layer variant in an attempt to avoid overfitting with relatively limited data) was selected as the baseline performance indicator. Although Inception v4 was reported to have slightly higher performance than ResNet, ResNet was chosen for further examination. This choice was primarily motivated by ResNet's comparatively ubiquitous usage across many applications [24]. EfficientNet, specifically its most lightweight B0 variant, will also be further examined to verify its high performance and computational efficiency, which is of particular interest in project part 2's embedded context.

NOTE: LFW is commonly used for **face verification rather than classification** in literature. Despite this, the brief (especially for project 2) seems to imply a classification task. Consequently, **I will evaluate the face classification performance, not pair matching performance.**

The testing methodology employed is as follows:

1. **Input Parsing & Data Preparation**
   The top 10 classes in the LFW database are parsed, and their corresponding labels are associated. The colour channel ordering is switched to RGB from BGR to match the models' expected channel ordering. I did consider reducing the massive class imbalance via data augmentation or cutting samples. However, I decided against this to 1) eliminate unintentional biases due to the specific data augmentation technique and 2) maintain as much data as possible to prevent overfitting to a limited dataset, especially for deeper models.

2. **Preprocessing**
   Different architectures can expect differing input resolutions. In our case, VGG, ResNet and EfficientNet-B0 all expect 224x224 images by default. To achieve this target resolution, we centre crop our 250x250 images down to 224x224. Using a centre crop minimises the chances of clipping the face as the faces are centre aligned.
   We also apply each architecture's preprocessing steps so as not to be biased towards any single model. For VGG and ResNet, this involves subtracting the ImageNet mean from our pixel values. For EfficientNet, this means a normalisation followed by Standardisation using the ImageNet mean and standard deviation.

3. **Pre-trained ImageNet Weights**
   We naturally require a lot of data to train and test deep neural networks such as those under test. Our top 10 LFW classes provide just 1456 samples, which risks overfitting if trained from scratch. This is doubly so due to their unbalanced nature regarding samples per class. Methods to supplement this, such as data augmentation, also risk intentionally introducing biases towards a model. Instead, we will load the pre-trained ImageNet weights and fine-tune the model.

4. **Fine Tuning with a New Classification Head.**
5. To keep "artificial" inter-architecture variances at a minimum, we aim to keep the original classification head of the models, with just the output SoftMax layer modified to output our ten classes. For VGG, this meant two fully connected layers; for ResNet and EfficientNet, it meant global average pooling before the SoftMax layer. For EfficientNet, there was also a dropout layer before the final SoftMax. I explicitly chose not to change hyperparameters related to the layers to maintain consistency across the models and reduce the introduction of unnecessary architectural bias. The intent is for the comparison between models to focus more directly on their inherent structural differences rather than on potentially confounding effects from overly specialised hyperparameter tuning.

Early stopping was employed with a maximum epoch count of 50, which was experimentally found to give enough time for convergence (or, more likely, triggering the early stopping callback with patience of 3). The splits were also performed with stratification to prevent further adverse effects from the unbalanced dataset.

6. Hyperparameter Tuning

We are left with two hyperparameters we must tune. 1) the batch size, and 2) the learning rate. A simple iterative grid search was used to narrow down the optimal values for each model.

IV. Performance Evaluation & Discussion

A. Performance

As we only have ten classes, comparing top-5 error rates to those of ImageNet, which possesses 1000 classes, is misleading. Instead, I chose accuracy, loss and the Area under the Receiver Operating Characteristic Curve (AUC) as the primary evaluation metrics. The latter was specifically chosen to give fairer representation to underrepresented classes (e.g. John Ashcroft with 53 images vs George W Bush's 530).
The accuracy, loss and AUC can be seen in Table 1.

TABLE I. SUMMARY OF EVALUATION METRICS

Model	Evaluation Metrics		
	Test Accuracy (Unweighted)	*Test Loss (Unweighted)*	*AUC (Macro Avg.)*
VGG-16	95.43%	0.1914	0.9978
ResNet-152	99.54%	0.0083	1.0000
EfficientNet-B0	99.54%	0.0220	0.9999

All three models achieved good performance, with VGG-16 performing noticeably worse (~4.1%) than ResNet-152 and EfficientNet-B0, which is also supported by literature [7], [12].

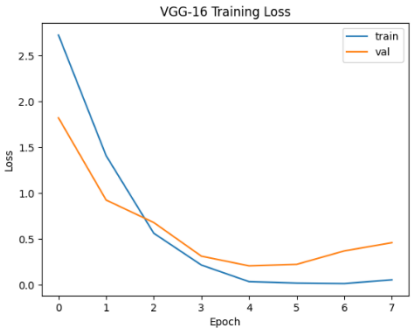

Figure 5: Training loss curve for VGG-16

Interestingly, VGG-16 rapidly started overfitting after ~4th epoch, which was not seen with the other models. This is illustrated in Figure 5. This may suggest that VGG is more susceptible to overfitting, given unbalanced datasets and a massive reduction in the number of target classes (down to 1000 in ImageNet). It also had the highest parameter count (~138 million parameters), which may have also contributed to the quick overfitting.

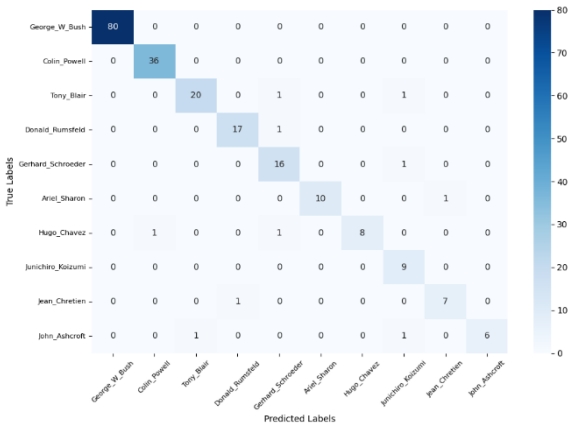

Figure 6: Confusion matrix for VGG-16

The confusion matrix for VGG-16 is seen in Figure 6. There were no clear trends within the misclassifications, with evenly distributed misclassifications across most classes.

ResNet-152 and EfficientNet-B0 shared very similar performance profiles. They achieved identical prediction accuracies, but ResNet-152 achieved lower loss, suggesting that ResNet-152 learned a more discriminatory decision function. Confusion matrices reveal that both models only got a single inference incorrect in the training set of 219. Both instances are shown below, alongside an actual photo of the predicted label in Figures 7 and 8.

Figure 7: Tony Blair's image misclassified by ResNet (left) and a sample from the predicted class (right).

Figure 8: Donald Rumsfeld's image misclassified by EfficientNet-B0 (left) and a sample from the predicted class (right).

As we can see, none are particularly egregious. A clear trend of misclassifications could not be found when testing beyond the training set within confusion matrices, suggesting both are robust in their classifications.

The training loss curves were nominal for both models, and can be seen in Figure 9.

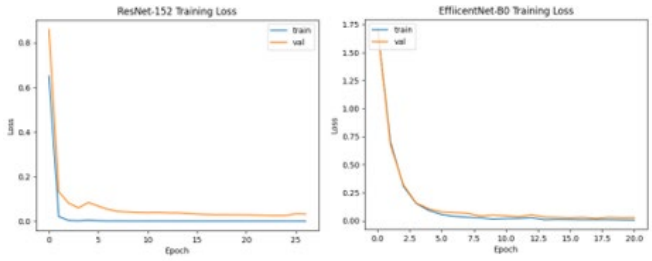

Figure 9: Training loss curves for ResNet-152 (left) and EfficientNet-B0 (right).

It must be stressed, however, that EfficientNet-B0, the base model for the EfficientNet architecture, achieves almost identical performance to ResNet-152 with only 5.3 million parameters compared to ResNet's 60 million – an 11x reduction in parameters. This also translates to less computational complexity, with only 0.39 billion FLOPs required compared to ResNet's 11 billion (~28x less), which was also noticeable during training. This may suggest a layer depth of 152 was too high for our classification task, and a smaller ResNet model more comparable to EfficientNet-B0 (in terms of parameter count), like ResNet-50 or ResNet-32, may be a better fit. Although parameter count is not a perfect analogue for representational power or computational complexity, it may be interesting to study further, given more time. Furthermore, VGG's performance under more ideal and balanced dataset environment may also be interesting to study further.

To that end, this qualitative and quantitative analysis suggests that **EfficientNet-B0** may be the most promising candidate to carry forward facial recognition tasks in an embedded context out of the five models examined in this report.

REFERENCES

[1] Itema, "Face Recognition: A Literature Review," ITEMA 2024. Accessed: Sep. 15, 2024. [Online]. Available: https://www.itema-conference.com/face-recognition-a-literature-review/

[2] R. Rossi, M. Lazarini, and K. Hirama, "A Systematic Literature Review on the Accuracy of Face Recognition Algorithms," *EAI Endorsed Trans. Internet Things*, vol. 8, p. e5, Sep. 2022, doi: 10.4108/eetiot.v8i30.2346.

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Apr. 10, 2015, *arXiv*: arXiv:1409.1556. doi: 10.48550/arXiv.1409.1556.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 10, 2015, *arXiv*: arXiv:1512.03385. doi: 10.48550/arXiv.1512.03385.

[5] C. Szegedy *et al.*, "Going Deeper with Convolutions," Sep. 16, 2014, *arXiv*: arXiv:1409.4842. doi: 10.48550/arXiv.1409.4842.

[6] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," Aug. 23, 2016, *arXiv*: arXiv:1602.07261. doi: 10.48550/arXiv.1602.07261.

[7] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Sep. 11, 2020, *arXiv*: arXiv:1905.11946. doi: 10.48550/arXiv.1905.11946.

[8] "Papers with Code - VGG Explained." Accessed: Sep. 15, 2024. [Online]. Available: https://paperswithcode.com/method/vgg

[9] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," Feb. 23, 2014, *arXiv*: arXiv:1312.6229. doi: 10.48550/arXiv.1312.6229.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[11] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," Nov. 28, 2013, *arXiv*: arXiv:1311.2901. doi: 10.48550/arXiv.1311.2901.

[12] "ILSVRC2014 Results." Accessed: Sep. 15, 2024. [Online]. Available: https://image-net.org/challenges/LSVRC/2014/results

[13] R. Moradi, R. Berangi, and B. Minaei, "A survey of regularization strategies for deep models," *Artif. Intell. Rev.*, vol. 53, no. 6, pp. 3947–3986, Aug. 2020, doi: 10.1007/s10462-019-09784-7.

[14] "ILSVRC2015 Results." Accessed: Sep. 15, 2024. [Online]. Available: https://image-net.org/challenges/LSVRC/2015/results

[15] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 4685–4694. doi: 10.1109/CVPR.2019.00482.

[16] "Papers with Code - ResNet." Accessed: Sep. 15, 2024. [Online]. Available: https://paperswithcode.com/lib/torchvision/resnet

[17] "Papers with Code - GoogleNet." Accessed: Sep. 15, 2024. [Online]. Available: https://paperswithcode.com/model/googlenet

[18] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments," University of Massachusetts, Amherst, 07–49, Oct. 2007.

[19] *NVlabs/ffhq-dataset*. (Sep. 14, 2024). Python. NVIDIA Research Projects. Accessed: Sep. 15, 2024. [Online]. Available: https://github.com/NVlabs/ffhq-dataset

[20] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," Mar. 29, 2019, *arXiv*: arXiv:1812.04948. doi: 10.48550/arXiv.1812.04948.

[21] K. Panetta *et al.*, "A Comprehensive Database for Benchmarking Imaging Systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 509–520, Mar. 2020, doi: 10.1109/TPAMI.2018.2884458.

[22] "UTKFace," UTKFace. Accessed: Sep. 15, 2024. [Online]. Available: https://susanqq.github.io/UTKFace/

[23] D. Belcar, P. Grd, and I. Tomicic, "Automatic Ethnicity Classification from Middle Part of the Face Using Convolutional Neural Networks," *Informatics*, vol. 9, p. 18, Feb. 2022, doi: 10.3390/informatics9010018.

[24] "Papers with Code - Inception-v4 Explained." Accessed: Sep. 16, 2024. [Online]. Available: https://paperswithcode.com/method/inception-v4