Multi-class Dog Breed Classification

Oybek Khakimjanov, Wenbo Zhao, Kaiwen Deng
Department of Computer Science, Rice University, Houston, TX 77005
GitHub link: https://github.com/Vader-WenboZhao/COMP576_final_project
[ojk1, wz47, kd45]@rice.edu

Abstract

Image classification is a fundamental task for computer vision and is essential for deep learning in practical use. Convolutional Neural Networks (CNN) reduce the high dimensionality of the images without losing their information and have proved to be effective in visual interpretation.[5] In this project, we use Stanford University Dogs Dataset [3] to train multiple deep learning models and our own multi-layer model. The trained models are able to accept a picture of a dog and determine the breed of the dog. Then, we compare the performance of different models with a group of pictures of different breeds of dogs as test samples. In addition, we visualize the training results to make the comparison between the different models more direct and then analyze the advantages and disadvantages of different models used in the project.

1. Introduction

Image Classification is the task of associating one or more labels to a given image. Over the years, it has been one of the most essential tasks since the beginning of machine learning. With the occurrence and development of deep learning, this specific task has revolutionized and driven technological advancements in the most important fields in our lives, including healthcare, manufacturing, automobile, and more.

Although as simple as it seems, there still remain a lot of difficulties for this task. From the perspective of the dataset, the image classification dataset can have a large number of images; each image can have a high dimensionality and a lack of labeled data for different fields. In a real-world scenario, the task becomes complex given the complexity of the real matters. The main challenges from this perspective are the variations (sub-class) of one major class, the viewpoint of an object in an image, the occlusion of the object in an image, and how to select the major object for an image that contains multiple objects.

For this project, we are going to examine one of the challenges in the image classification task: how to classify the variations of one major class. We will use transfer learning techniques to fine-tune some of the most phenomenal CNN models in the image classification tasks and then do benchmarking and analysis on their performances. In the meanwhile, we try to design our own model to compare with these models to see its advantages and disadvantages.

2. Design and Specification

For this project, we will be using **Stanford Dogs** [3], [1] dataset presented by Stanford Vision and Learning Lab [1].

This dataset provides images of **120 breeds** of dogs from around the world. The contents of the dataset are the following:

- Number of categories: 120
- Total number of images: 20,580
- Annotations: Class labels, Bounding boxes
- Dimensions: Only images of 200 x 200 pixels or larger are kept in the dataset

The dataset will be split into training (65%), testing(30%), and validation(5%) sets. In total, this dataset represents the most comprehensive and diverse dataset for breed classification. Each image is annotated with an object class label see Figure 1.

This dataset is particularly challenging due to a variety of reasons. First, being a fine-grained classification problem, there is little inter-class variation. For example, certain breeds of dogs may be very akin to one another and share similar characteristics and colors. Second, most of the images in a dataset contain humans and other unrelated objects, thus, increasing the noise caused by background variation. Unlike other fine-grained visual datasets, this dataset has a larger number of images per category, and thus it can be used for accurate testing of different learning and optimization algorithms.

[1] http://vision.stanford.edu/aditya86/ImageNetDogs/

n02094433-yorkshire_terrier (36)
n02115913-dhole (118)
n02097130-giant_schnauzer (46)
n02111129-leonberg (103)
n02113624-toy_poodle (113)
n02113978-mexican_hairless (116)
n02102973-irish_water_spaniel (70)
n02088094-afghan_hound (9)
n02088632-bluetick (13)

Figure 1. Sample from Dogs Breed Dataset

3. Model

In order to do benchmarking, we plan to compare several Computer Vision models plus our own model in this project.

3.1. VGG-16

The VGG-16 [6] model is one of the most popular pre-trained models for image classification tasks. Brought up in 2014, it beat the top-performance model AlexNet [4] and still keeps high-level performances today. This model is sequential in nature and uses a lot of 3×3 filters. At each stage, these small filters are used to reduce the number of parameters followed by the ReLU activation function. Compared to AlexNet, VGG-16 is a much larger model with a huge amount of parameters, which makes it much slower when training. But in general, this model is simple, intuitive, and one of the top players in image classification tasks. The architecture of VGG-16 is shown in Figure 2.

3.2. Resnet

The Resnet152(Residual Net) [2] is not the first model of Resnet but it offers great performance on image classification tasks.

There was a major problem for the deep learning model before Resnet: as the model went on to become deeper, the accuracy became poorer. So this problem became the main motivation of the Resnet model. Surprisingly, the Resnet model can also tackle the Vanishing Gradient issue.

In the Resnet model, after a 7×7 convolutional layer and a max-pooling layer, there are 4 similar layers with different filter sizes, but all of them use 3×3 operation. The architecture of Resnet is shown in Figure 3.

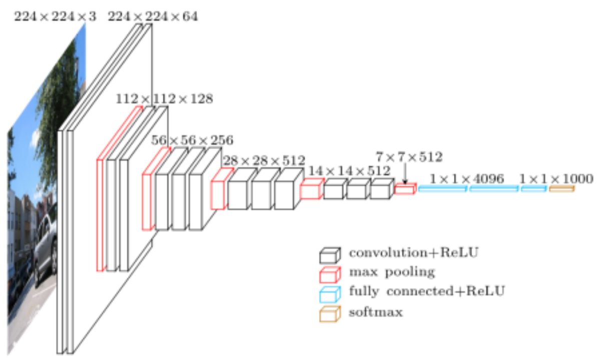


Figure 2. VGG-16 Model

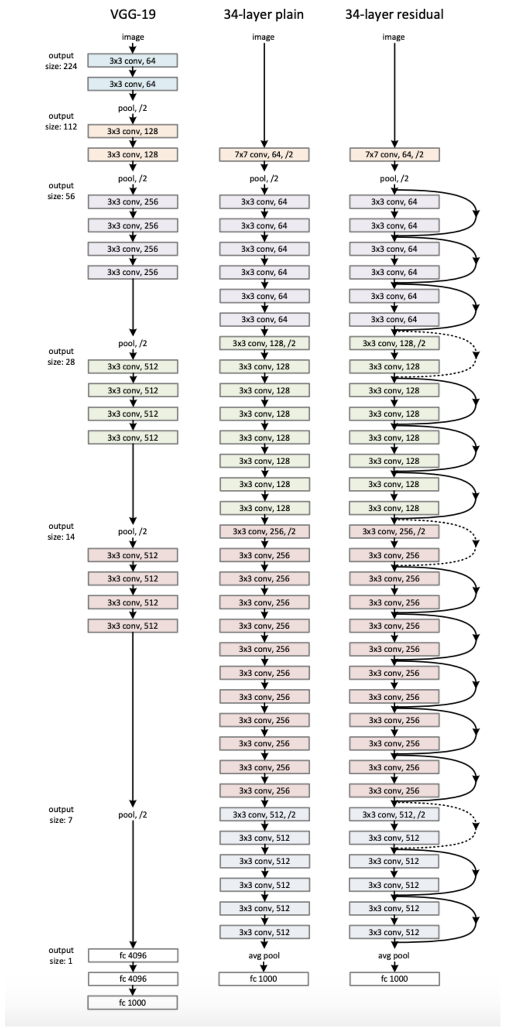
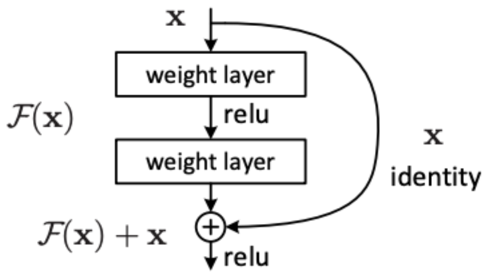


Figure 3. VGG-16 vs Resnet

However, the key point in this model is: after every two convolution layers, the model bypasses the layer in between. These skipped connections are called "identity shortcut connections". As a result of using this, the model is able to pass the original information to the deep convolutional layers. The building block of Resnet is shown in Figure 4.



x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4. A block in Resnet

3.3. Inception

The Inception model [7], as known as GoogLenet, was another astonishing image classification model in 2014. Compared to VGG and AlexNet, it is much smaller in the size with only 7 million parameters and a relatively low error rate.

The fundamental part of the Inception model is the Inception Module. Simply speaking, this module performs convolution with different filter sizes or max pooling on the input, then concatenates the result for the next Inception Module. The Inception Module is shown in Figure 5.

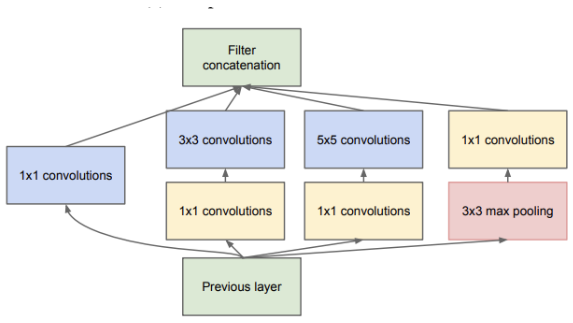


Figure 5. The Inception Module

With that design, the Inception model can achieve astonishing results with only 22 layers. And the newest InceptionV3 introduces batch normalization and RMSProp optimizer to the model and adds more factorization, which largely increased the accuracy and made the model less complex. The architecture of Inception is shown in Figure 6.

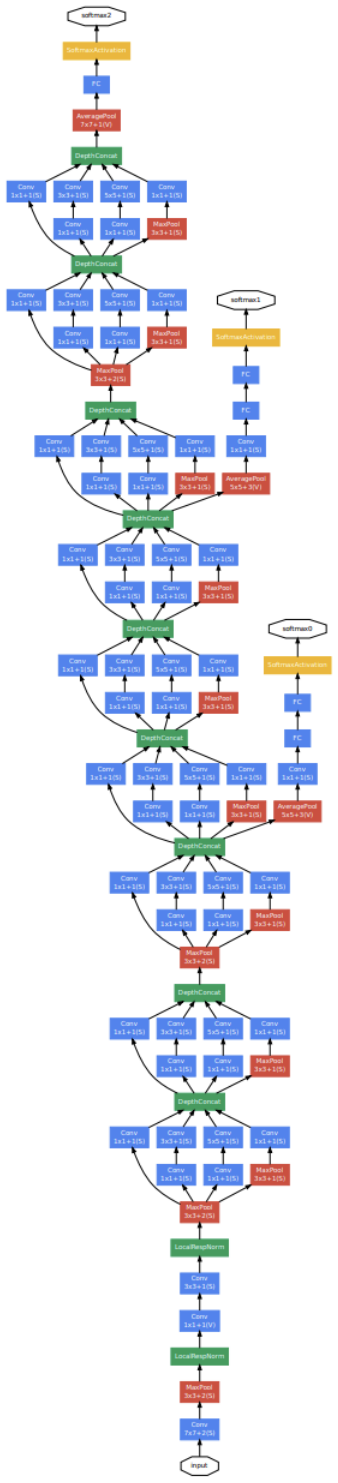


Figure 6. The Inception Model

3.4. EfficientNet

The EfficientNet model [8] proposed by Google is another major progress in the Computer Vision field. In this paper, they proposed Compound Scaling - a new Scaling method that proposes that if we scale the dimensions by a fixed number and do uniformly, the model will achieve much better performance. The architecture of EfficientNet is shown in Figure 7.

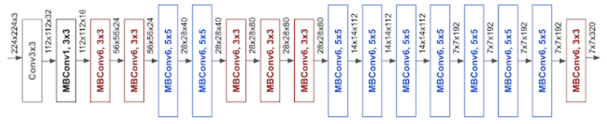


Figure 7. The EfficientNet Model

3.5. Our Model

For our model for this project, we decide to select two of the four existing models - VGG16 and Resnet152 and combine them into one model. To combine them and make our model with the correct number of outputs, we first concatenate the results from these two models, then use a linear layer as a classifier from the length of the concatenation to the number of classes. In the end, we use the Relu function as our activation layer. The architecture of our model is shown in Figure 8.

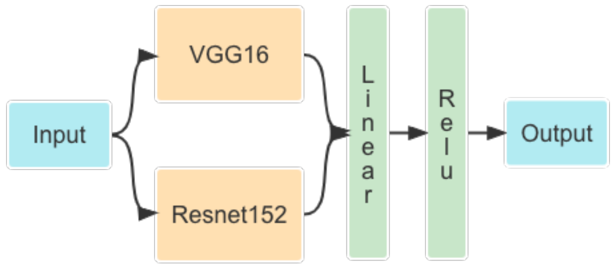


Figure 8. Our Model

4. Experiments and Results

4.1. VGG-16

This model comprises 16 CNN weight layers, a total of 21 layers. It takes input sensor size as 224×224 with 3 RGB channels and has a large number of hyper-parameters, 3×3 filters with stride 1. Same padding and max pool layer with 2×2 filters of stride 2.

During the training process, we use cross entropy as our loss function, Adam optimizer as our optimizer, a batch size of 128, a learning rate of 0.0001, and pre-trained parameters offered by Pytorch. The loss over the training process is shown in Figure 9 and the accuracy over the training process is shown in Figure 10.

As a result, the fine-tuned VGG-16 model achieves an overall accuracy of 82.3% on the test set.

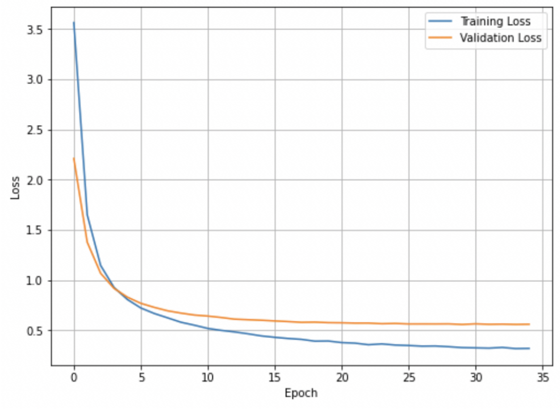


Figure 9. VGG-16 loss

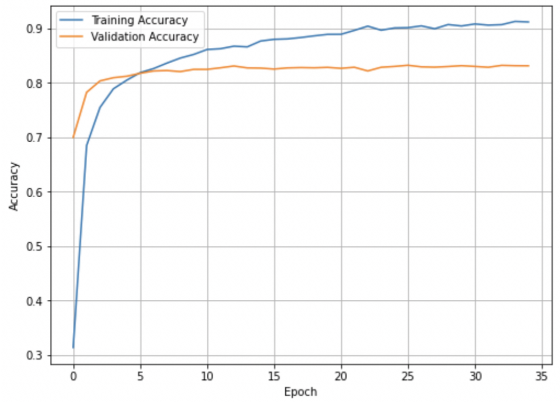


Figure 10. VGG-16 accuracy

4.2. Resnet

All convolutional layers apply the same convolutional window of size 3×3. Only one max-pooling layer with a pooling size of 3×3 and a stride of 2 is applied after the first layer. The average pooling layer is applied to replace fully connected layers. It uses ”identity shortcut connections” in blocks.

During the training process, we use cross entropy as our loss function, Adam optimizer as our optimizer, a batch size of 128, a learning rate of 0.0001, and pre-trained parameters offered by Pytorch. The loss over the training process is shown in Figure 11 and the accuracy over the training process is shown in Figure 12.

As a result, the fine-tuned Resnet model achieves an overall accuracy of 83.1% on the test set.

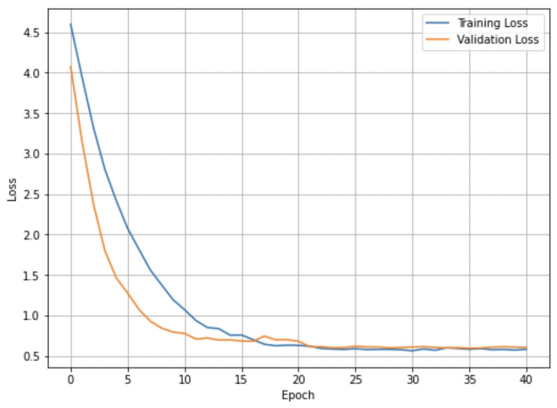


Figure 11. ResNet152 loss

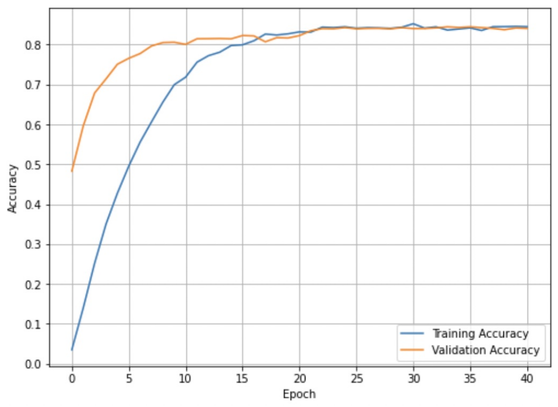


Figure 12. ResNet152 accuracy

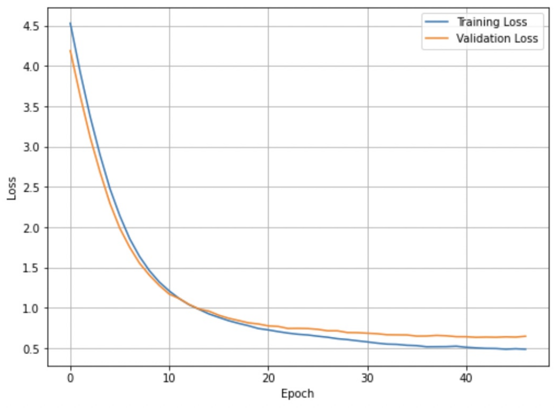


Figure 13. Inception loss

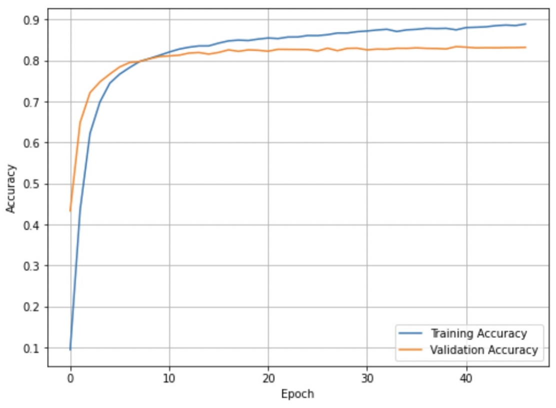


Figure 14. Inception accuracy

4.3. Inception and EfficientNet

These two models are both much smaller in size and have relatively low error rates. While:

- Inception: performs convolution with different filter sizes or max pooling on the input, then concatenates the result for the next module.

- EfficientNet: scale the dimensions by a fixed number and do uniformly, the model will achieve much better performance.

During the training process for Inception, we use cross entropy as our loss function, Adam optimizer as our optimizer, a batch size of 128, a learning rate of 0.0001, and pre-trained parameters offered by Pytorch. The loss over the training process is shown in Figure 13 and the accuracy over the training process is shown in Figure 14.

As a result, the fine-tuned Inception model achieves an overall accuracy of 82.5% on the test set.

During the training process for EfficientNet, we use cross entropy as our loss function, Adam optimizer as our optimizer, a batch size of 128, a learning rate of 0.0001, and pre-trained parameters offered by Pytorch. The loss over the training process is shown in Figure 15 and the accuracy over the training process is shown in Figure 16.

As a result, the fine-tuned Inception model achieves an overall accuracy of 79.1% on the test set.

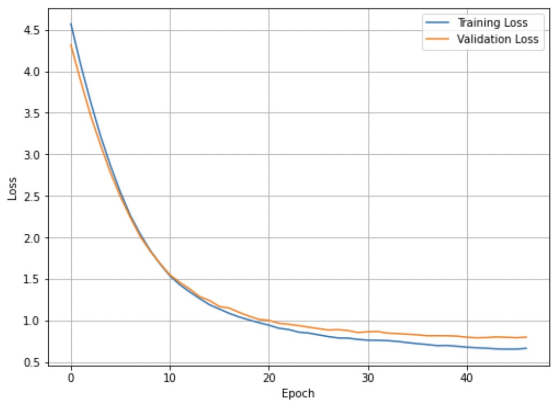


Figure 15. EfficientNet loss

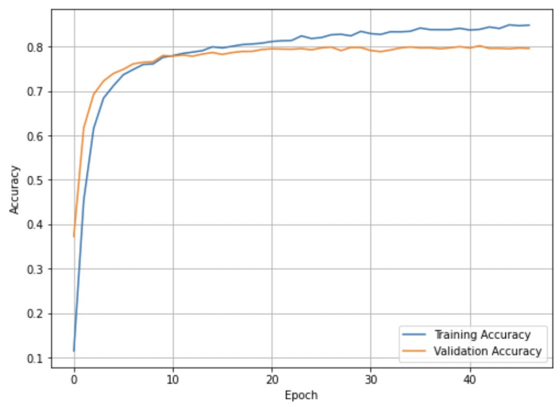


Figure 16. EfficientNet accuracy

4.4. Our model

As mentioned before, our model uses a combination of VGG16 and Resnet. After the concatenation of these two models, we use a 376 x 128 linear layer as the classifier to let the model make the right prediction that matches the number of classes in the dataset. In the end, we use the Relu function as our activation layer.

During the training process for our model, we use cross entropy as our loss function, Adam optimizer as our optimizer, a batch size of 128, a learning rate of 0.0001, and pre-trained parameters for both VGG16 and Resnet offered by Pytorch. The loss over the training process is shown in Figure 17 and the accuracy over the training process is shown in Figure 18.

As a result, the fine-tuned Inception model achieves an overall accuracy of 82.1% on the test set.

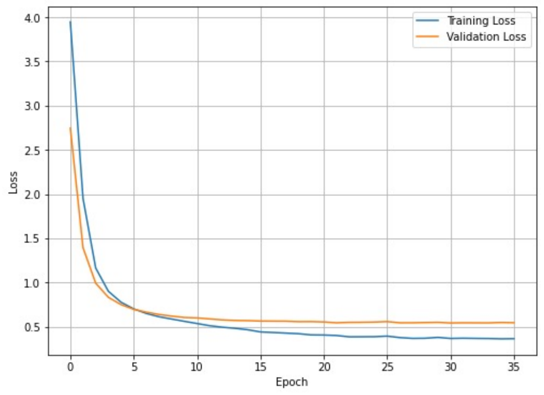


Figure 17. Our model loss

5. Potential Issues

While interesting, image classification for this dataset can be a very computing-demand task for a machine. Several challenges particular to the image classification problem have been noted by us.

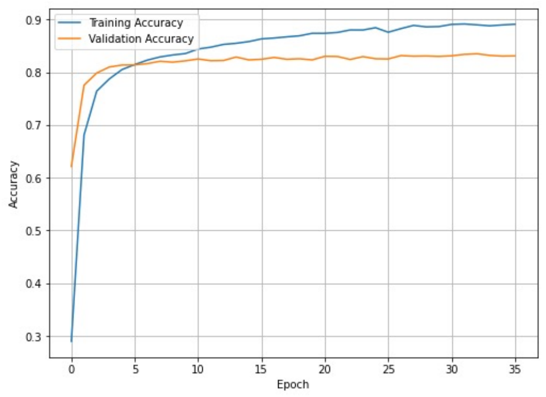


Figure 18. Our model accuracy

5.1. Challenges

- It is a fine-grained categorization problem that contains a little inter-class variation.
- here is very large intra-class variation, i.e., dogs within a class can have different ages, poses, colors, etc.
- Large proportion of the images in the dataset contains humans or more than one dog breed and are taken in homemade environments leading to greater background variation.
- Limited computational resources lead to a relatively long training time.
- Some dog breeds are initially having much more images than others.

5.2. Methods Revised

The methods revised contain:

- VGG-16 (CNN), ResNet152 (CNN), Inception EfficientNet
- Used transfer learning technique to avoid training complex models from scratch to achieve better results in less time.

5.3. Analysis Further Research

- Evenly distributes images for each breed.
- Using image augmentation to increase the size of the training set.
- Create Deeper Neural Network Topology.
- Use better resampling techniques like K-fold cross-validation.
- Benchmark and experiment with different models on the dogs dataset.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[5] N. Lang. Using convolutional neural network for image classification, 2021.

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[8] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.