ARCHITECTURAL STYLE CLASSIFICATION

A PREPRINT

Cheng Fei
Cornell Tech
New York, NY
cf482@cornell.edu

Tianyu Lou
Cornell Tech
New York, NY
tl838@cornell.edu

Wenjie Wang
Cornell Tech
New York, NY
ww523@cornell.edu

January 5, 2024

ABSTRACT

This research paper introduces a novel architecture style classification model aimed at discerning various architectural styles based on their visual characteristics. As individuals explore new environments, encountering buildings of historical or unique significance often sparks curiosity about their architectural styles and the narratives they encapsulate. However, recognizing and cataloging diverse architectural styles can be challenging for humans due to their sheer abundance and complexity. In response to this challenge, our study proposes the development of an architecture style classification model capable of real-time recognition. This model holds promising applications, particularly for tourists who can capture images of buildings and promptly identify their architectural styles. The primary objective is to create a system adept at recognizing architectural styles across different historical eras and various architects' designs. By employing advanced visual recognition techniques, our model aims to contribute to the seamless integration of technology into architectural appreciation, providing an accessible tool for enthusiasts, historians, and tourists alike. The potential impact of such a model extends beyond individual curiosity, offering a practical solution for instant architectural style identification in diverse scenarios.

Keywords Architectural Style · Convolutional Neural Network · ResNet

1 Introduction

The rich tapestry of architectural styles that grace our cities and landscapes stands as a testament to the diverse creativity of human civilization. However, for individuals navigating urban environments or exploring historical sites, the challenge of identifying and understanding the architectural styles of buildings remains a significant barrier to a deeper appreciation of our built heritage. This paper introduces ArchVisioNet, an innovative architecture style classification model designed to address this challenge by leveraging advanced visual recognition techniques. As technology continues to intersect with various facets of our lives, the fusion of artificial intelligence and architectural appreciation takes center stage in ArchVisioNet. This model not only aims to unravel the complexities of architectural styles in real-time but also seeks to empower users, especially tourists, enthusiasts, and historians, with a tool capable of providing instantaneous insights into the historical eras and architects behind diverse architectural designs. Through a synthesis of cutting-edge technology and a passion for architectural heritage, ArchVisioNet represents a pioneering step towards democratizing architectural knowledge and fostering a deeper connection between individuals and the rich architectural tapestry that surrounds them.

2 Background

2.1 Context

In comparing our project on architectural style classification using Convolutional Neural Networks (CNN) with significant prior works in the field, we observe both continuities and advancements. Our project draws inspiration from

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton's groundbreaking work on ImageNet Classification with Deep Convolutional Neural Networks, which revolutionized the approach to deep learning in image classification. While their work laid the foundation for using deep CNNs, our project extends these principles specifically to architectural styles, emphasizing the nuanced differences and unique characteristics inherent in this type of imagery.

Similarly, the work of Kaiming He and his colleagues on Deep Residual Learning for Image Recognition, particularly their development of ResNet, influenced our model's architecture. The deep residual learning framework they introduced allowed us to build deeper networks, crucial for capturing the intricate details necessary in architectural classification. However, our project diverges in its application and fine-tuning, tailoring the model to recognize and differentiate between architectural styles rather than the broader object recognition in He et al.'s study.

Furthermore, our project benefits from advances in transfer learning and data augmentation, concepts not fully explored in these earlier works. We leverage pre-trained networks and sophisticated data augmentation techniques to address the unique challenges of architectural image classification, such as varied lighting conditions and perspectives. These advancements demonstrate our project's evolution beyond the foundational work, focusing on specialized applications of CNNs in a domain-specific context.

2.2 Architectural Styles

Architectural styles refer to the distinct characteristics and features that define the appearance and structure of buildings across different periods and regions. These styles are categorized based on elements like form, method of construction, building materials, and regional cultural influences. Common examples include Gothic, Romanesque, Baroque, Victorian, Modernist, and Postmodernist styles. The classification of architectural styles is crucial in the fields of art history, urban planning, and conservation, aiding in the understanding of cultural and historical contexts of architecture.

2.3 Dataset

The initial dataset consists of 25 distinct architectural styles, each represented by 400 images on average. We generate the dataset from the ground up by extracting images from online sources like Google Images and assigning each image a corresponding label manually. In order to increase the quality of scraped datasets, we optimize the python crawler to filter out noisy datasets. Generally, these architectural styles are among the most common ones people can find in their daily lives, which ensures the applicability and feasibility of the model.

Subsequently, we undertake data preprocessing tasks, ensuring that all images are standardized in size without sacrificing any key features, and noisy information is removed which consists of reorientation and rescaling. Then, we perform data augmentation to produce a large batch of data in order to optimize the model accuracy. This step consists of adjusting the sharpness, color, brightness, and contrast.

Figure 1: Dataset Visualization

2.4 Convolutional Neural Network

CNNs are a class of deep neural networks, primarily used in processing data with a grid-like topology, such as images. They are exceptionally adept at recognizing patterns in images, making them ideal for tasks like image classification, object detection, and even style transfer. A CNN automatically detects important features without any human supervision, using a series of convolutional and pooling layers. Each layer in a CNN transforms the input data to enable the network to learn more complex features in deeper layers, making it a powerful tool for image analysis.

2.5 Transfer Learning

Transfer learning is a machine learning technique where a model developed for one task is reused as the starting point for a model on a second task. In the context of CNNs, this often means using a pre-trained network that has already learned a comprehensive set of image features on a large dataset (like ImageNet). The pre-trained network can then be fine-tuned with a smaller dataset specific to the target task, such as architectural style classification. This approach is

beneficial because it significantly reduces the computational resources and time required to train a deep neural network, while often enhancing performance, especially when the available dataset for the new task is relatively small.

This model leverages the pattern recognition power of CNNs and the efficiency of transfer learning. It can identify and classify various architectural styles from images, which can be an invaluable tool in fields like cultural heritage conservation, urban development, and even education in architectural history.

3 Method

3.1 Data Preprocessing

Data preprocessing prepares the dataset for the training task. We perform two major steps in order to expedite the training procedures.

Normalization: For image data, normalization usually involves adjusting the pixel values. Since pixel values range from 0 to 255, a common practice is to scale these values to a range of 0 to 1. This is done by dividing all pixel values by 255. This scaling helps in speeding up the convergence by reducing the variability in the input data. Normalization helps in optimizing the training speed by making the training process more stable and efficient, and by preventing issues like vanishing or exploding gradients in deep learning models.

Autotuning: In distributed training scenarios, autotuning can optimize the allocation of computational resources (like CPUs and GPUs) to different parts of the model or data, ensuring that the training process is as fast as possible. Autotuning is particularly beneficial in reducing the manual effort involved in hyperparameter tuning and in achieving faster convergence with optimal resource utilization. It helps in ensuring that the model trains efficiently without the need for extensive manual experimentation.

3.2 Data Augmentation

Data augmentation is a crucial step in improving the performance and robustness of machine learning models, particularly in image classification tasks like Architectural Style Classification. Here are some data augmentation techniques we used in the model:

Rotation: This is especially useful as buildings and architectural photographs can be captured from different angles.
Flipping: This can be particularly useful for symmetrical structures.
Scaling: This is important because architectural photos can be taken from various distances, altering the apparent size of the structure in the image.
Cropping: Random cropping of images can simulate partial views of buildings, a common scenario in architectural photography, especially in urban settings where full views are often obstructed.

Brightness Adjustment: Architectural photos can be taken in various lighting conditions, from bright sunlight to overcast skies.
Contrast Adjustment: This trains the model to be more adaptable to variations in image quality and style representation.
Color Adjustment: Adjusting the color properties of the images (like saturation, hue, etc.) can help the model in recognizing styles under different color conditions and photographic styles.
Perspective Adjustment: This helps the model understand architectural styles in a three-dimensional context.

Adding Noise: Introducing a small amount of random noise (like Gaussian noise) can improve the robustness of the model against grainy or low-quality images.
Adding Filter: Using various filters (like edge enhancement, sharpening, etc.) can help in highlighting or suppressing specific features of architectural designs, aiding in better feature extraction by the CNN.

3.3 Integration of Residual Networks in CNN

Skip Connections: The core idea behind ResNet is these skip connections that allow the input to "skip over" some layers. These connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. Technically, they bypass or 'skip' one or more layers and connect to a later layer in the network. This approach allows the model to carry forward features from previous layers without the risk of degradation during deeper layers.

Applications: In our project, we utilize ResNet due to its ability to efficiently train deeper networks, a necessary requirement for the complex task of architectural style classification. The intricate details and subtle nuances of architectural styles demand a network capable of extracting deep, sophisticated patterns. ResNet, with its deep structure

enhanced by skip connections, ensures that our model learns these intricate patterns without suffering from the vanishing gradient problem, thus leading to more accurate and reliable classifications.

4 Setup

The development of the CNN model goes through four major stages: model selection, initial model construction, parameter fine-tuning, and performance evaluation.

4.1 Phase 1: Model Selection

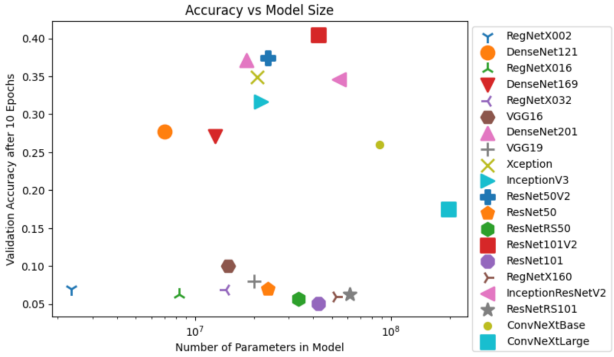


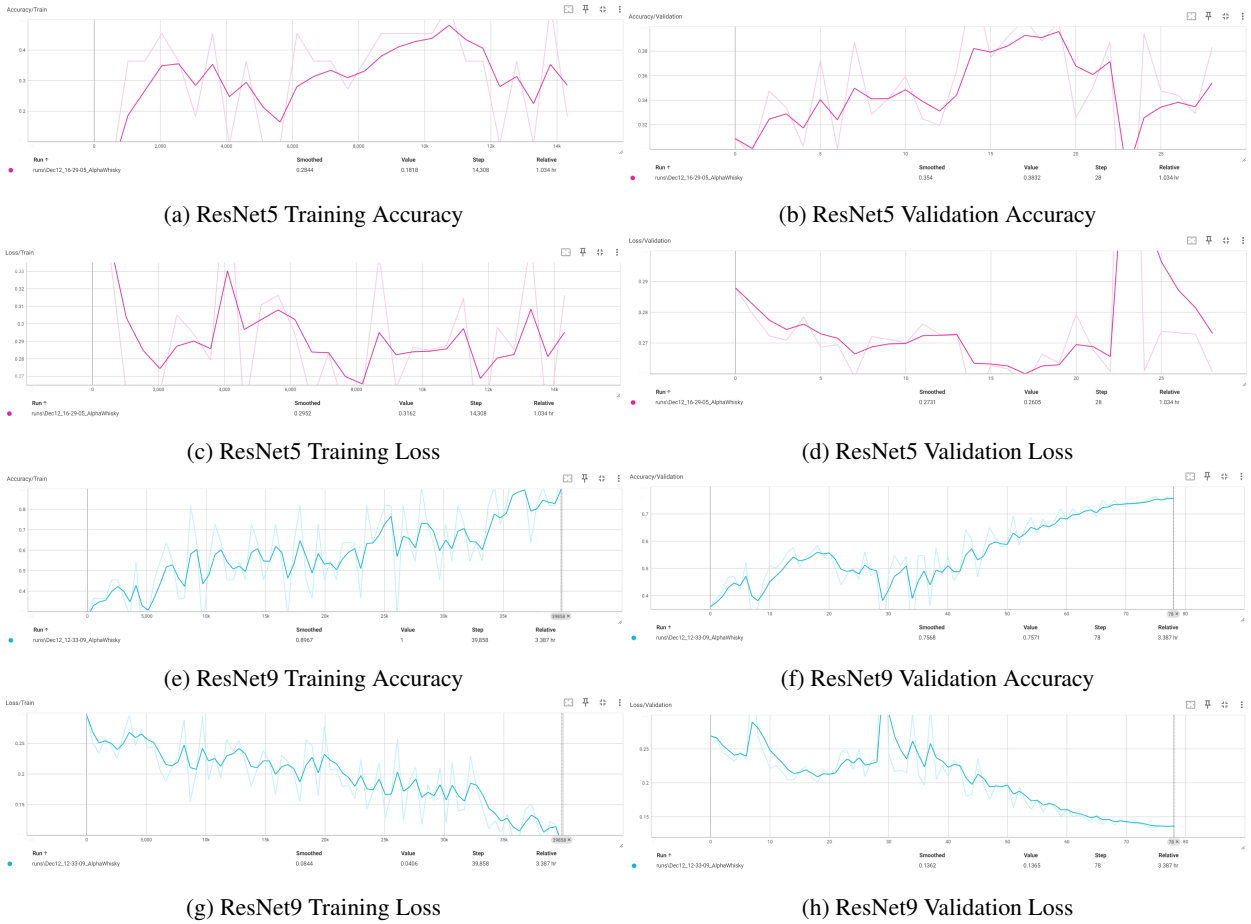
Figure 2: Different Model Accuracy

To identify the most suitable CNN architecture for classifying architectural images, we conducted a thorough evaluation of various Convolutional Neural Network (CNN) architectures to determine the most suitable model for classifying architectural images in our initial phase of research. This evaluation encompassed a diverse range of CNN models, including well-known architectures such as VGG, Inception, and ResNet. Each of these models was assessed for their potential effectiveness in our specific application. Ultimately, the ResNet architecture was selected for its advanced residual learning framework. This framework is particularly beneficial as it effectively combats the vanishing gradient problem that often plagues deep neural networks, ensuring consistent learning even as the network depth increases. The selection criteria were not solely based on architectural features; we also considered critical performance metrics. These included the accuracy of the model and the speed of convergence during preliminary testing. Another key factor in our decision was the model's ability to handle deep architectures without significant performance degradation. ResNet's capability to maintain robust performance in deep network structures, without losing efficiency or accuracy, made it the standout choice for our complex image classification task.

4.2 Phase 2: Initial Model Development and Overfitting Challenges

Build ResNet 9 from scratch: The objective of this phase of our research was to construct and refine a Convolutional Neural Network (CNN) model using PyTorch, tailored specifically for the effective classification of various architectural styles. We build a ResNet 5 from the ground up to test the baseline performance of the model. According to the performance plots, the ResNet 5 model underfits the dataset and converges quickly. Hence, we build a more expressive model, ResNet 9, to improve the model's training accuracy. However, ResNet 9 turns out to overfit the current dataset.

Causes of overfitting: Normally, a model overfitting a dataset is caused by two conditions – either the model is too expressive or the dataset is too small. Hence, we come out with two possible approaches to resolving this issue – regularizing the model or fetching more data. We first apply the data augmentation techniques, including rotation, scaling, and horizontal flipping, to enrich our training dataset. Data augmentation basically enable the model to identify key features from abnormal images, such as images taken in a dark setting or taken from a skewed perspective. The model trained on the augmented dataset performs better than the last version. Then, to streamline the model's focus and enhance its accuracy, we simplified the classification categories by consolidating them. This step involved merging similar architectural styles into broader categories, decreasing the categories to classify from 25 to 10, thereby reducing the complexity and potential ambiguity of the classification task. However, since augmented data share a large portion of key features with the original dataset, the model still reaches a performance bottleneck. Hence, we adopt the regularization technique.



(a) ResNet5 Training Accuracy

(b) ResNet5 Validation Accuracy

(c) ResNet5 Training Loss

(d) ResNet5 Validation Loss

(e) ResNet9 Training Accuracy

(f) ResNet9 Validation Accuracy

(g) ResNet9 Training Loss

(h) ResNet9 Validation Loss

Figure 3: ResNet5 and ResNet9 Training and Validation Results

Regularization: We manually add Dropout layers to allow the model to randomly set some fractions of the dataset to 0 at each update and prevent the parameters from co-adapting too much. Compared with its former performance plots, the regularized ResNet9 model produces a higher accuracy in both the training set and the cross-validation set. That is a good sign for the low bias and the low variance.

Pivot to transfer learning: Although we handled the over-fitting issue in the ResNet 9 model, there are two subsequent concerns. Compared with using a pre-trained model, it consumes a significant amount of time and data to train it from scratch. However, it is neither time-efficient nor resource-efficient to train the whole model from the ground up because we can borrow the general image classification capabilities from pre-trained CNN models. Hence, in the next stage, we import a larger pre-trained model and feed our specific dataset.

4.3 Phase 3: Fine-tuning and Addressing Validation Bottlenecks

In response to prior findings, we pivoted our strategy to include the implementation of transfer learning. This involved utilizing a pre-trained ResNet 18 model, which had been initially trained on a large and diverse dataset, such as ImageNet. The rationale behind this choice was that a model pre-trained on a vast and varied dataset would have already learned a substantial amount of features, which could be beneficial for our task. To tailor this model to our specific needs in architectural image classification, we conducted fine-tuning on the last few layers of the ResNet 18 model. This process involved retraining these layers with our architectural image dataset, allowing the model to adjust its learned features to our specific classification task. This method of integrating transfer learning not only addressed the overfitting issue but also leveraged the advanced learning that the model had already achieved, thus enhancing its performance on new, unseen data.

4.4 Phase 4: Final Model and Performance Evaluation

The concluding phase of our project was focused on two key objectives: finalizing the model after iterative refinements and comprehensively evaluating its overall performance. To achieve this, we employed a structured methodology encompassing both the finalization of the model and its subsequent performance assessment.

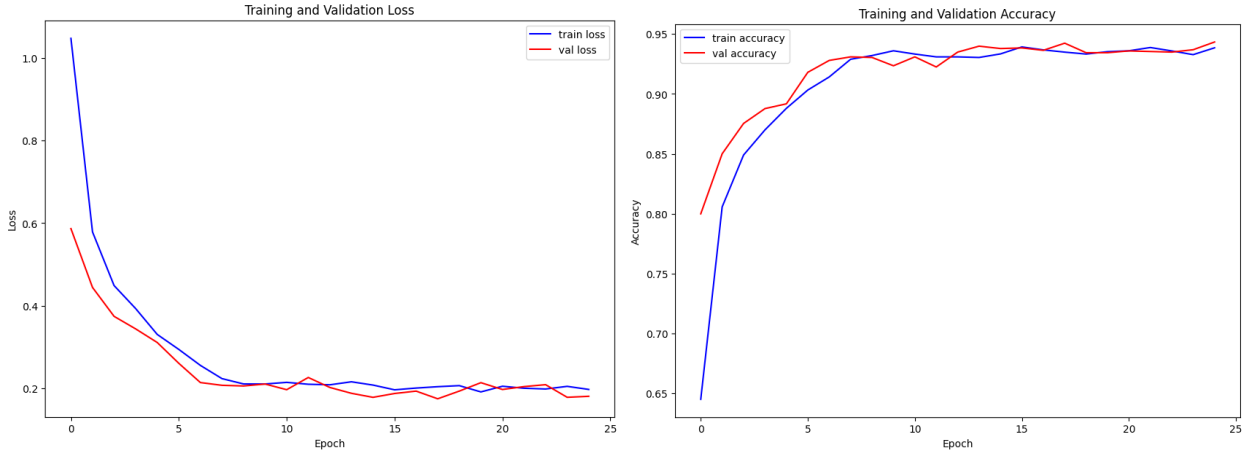
Finalization of the Model: Our journey of refinement and optimization culminated with the selection of the ResNet 18 model as our final model. This decision was informed by the model's performance during the previous phases, particularly its ability to effectively address the overfitting issues we encountered.

An integral part of this finalization process involved the meticulous optimization of hyperparameters. Parameters such as learning rate, batch size, and the number of epochs were fine-tuned to align with the performance metrics we aimed to optimize. This step was crucial to ensure that the model not only performed optimally in terms of accuracy but also operated efficiently.

Performance Evaluation: To rigorously assess the model's performance, we utilized a separate validation set that had not been used during the training phase. This approach was critical to objectively evaluate the model's ability to generalize and perform on new, unseen data.

A suite of metrics was employed to provide a holistic view of the model's performance. These included accuracy, which measures the proportion of correctly predicted instances; precision, indicating the proportion of positive identifications that were actually correct; recall, reflecting the proportion of actual positives that were correctly identified; and the F1 score, which provides a balance between precision and recall. These metrics together offered a comprehensive understanding of the model's strengths and areas for improvement, giving us a detailed picture of its overall effectiveness in classifying architectural styles. In our experiment, the F1 score is 0.93, the recall is 0.93, and the precision is 0.93. These key metrics prove that our model is good enough to identify the correct architecture labels in the picture.

In conclusion, this final phase was pivotal in solidifying the reliability and robustness of our CNN model, ensuring it not only achieved high accuracy but also maintained a balanced performance across various metrics. This rigorous evaluation process was fundamental in affirming the model's readiness for application in real-world scenarios.



(a) ResNet18 Training and Validation Loss

(b) ResNet18 Training and Validation Accuracy

Figure 4: ResNet18 Training and Validation Results

5 Conclusion

5.1 Result

As we conclude this project, it is evident that the development and refinement of the CNN model for architectural image classification has been both challenging and rewarding. The successful implementation of the ResNet 18 model, after navigating through various stages of optimization and problem-solving, demonstrates a significant achievement in the field of computer vision, particularly in the specialized area of architectural style classification. However, like all research endeavors, this project is not without its limitations and avenues for future exploration.

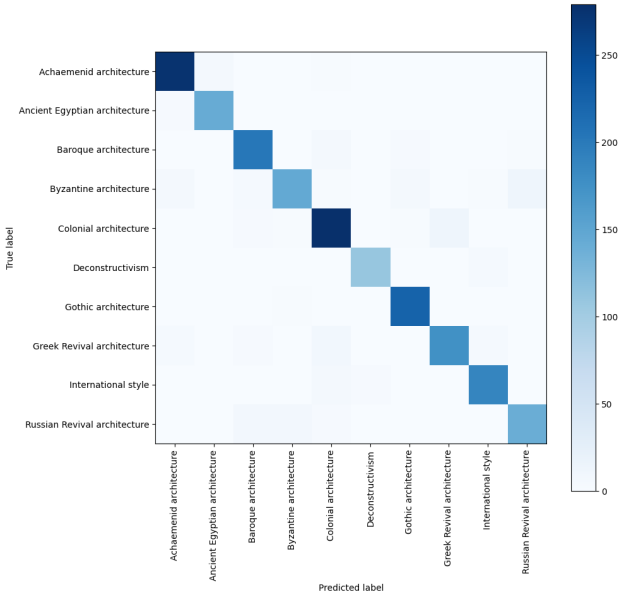


Figure 5: ResNet 18 F1-Score Confusion Matrix

5.2 Discussion and Future Works

One of the potential limitations of our current model is the reliance on the dataset used for training and validation. While extensive, there's always room for expanding the dataset to include a wider array of architectural styles, especially those from underrepresented regions or historical periods. Additionally, the current model may face challenges in adapting to drastically different architectural features not present in the training set.

Finally, adapting the model for real-time classification in mobile or web applications could significantly broaden its practical applications, making it a valuable tool for educational purposes, in architectural firms, or for cultural preservation projects.

In summary, the project stands as a testament to the potential of CNNs in specialized image classification tasks, setting a foundation for further research and development in this exciting field of computer vision.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[2] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[3] 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 (CVPR)*, 2015, pp. 1–9.

[4] F. Chollet, *Deep Learning with Python*. Manning Publications, 2017.

[5] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. MIT press Cambridge, 2016.

[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[7] 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 (CVPR)*, 2016, pp. 770–778.

[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[10] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6848–6856.