# Tiny-ImageNet Classification

Bhuvan Channagiri
channagiri.b@northeastern.edu

Gayathri Suresh
suresh.gay@northeastern.edu

Nikita Mandal
mandal.n@northeastern.edu

## I. Abstract

*This project explores the classification of the Tiny ImageNet dataset using various convolutional neural network (CNN) architectures and methodologies, including image augmentation and transfer learning. Starting with simple three-layer CNNs and progressing to more sophisticated deep neural networks with and without dropout, we assess their performance in handling complex image classification tasks. Subsequent experiments leverage pre-trained models like ResNet-50, highlighting the impact of deep architectures and ImageNet weights on performance. The project culminates in a comparative analysis that demonstrates the superior efficacy of deeper networks with dropout and transfer learning strategies, contributing to the ongoing discourse on optimizing CNN architectures for enhanced image classification accuracy.*

## II. Introduction

Image classification, a fundamental task in the field of computer vision, involves categorizing images into predefined classes based on their visual content. Deep learning, particularly through the use of Convolutional Neural Networks (CNNs), has transformed this domain by enabling the automatic learning of hierarchical feature representations directly from image data. CNNs, with their architecture inspired by the human visual system, excel at identifying complex patterns and nuances in visual inputs.

Our project focuses on the Tiny ImageNet dataset, a streamlined version of the larger and more diverse ImageNet dataset. Tiny ImageNet comprises 100,000 images across 200 classes, each image resized to 64x64 pixels, significantly smaller than the standard sizes used in the full ImageNet challenge. This reduction in size and scope provides a manageable yet challenging platform for deep learning models, making it an excellent medium for exploring the efficacy of various CNN architectures.

The exploration begins with fundamental CNN designs, starting with a basic three-layer network and progressively incorporating additional layers to examine how increased depth influences the network's learning capabilities. We introduce advanced features such as dropout to combat overfitting and improve the model's generalization abilities. Additionally, the project utilizes transfer learning strategies by employing weights from models pre-trained on the complete ImageNet dataset, which facilitates significant improvements in performance by leveraging pre-learned features.

## III. Background

Tiny ImageNet serves as a streamlined subset of the vast ImageNet database, tailored for training and testing image classification models within a more confined computational scope. This dataset comprises

200 diverse classes, each represented by 500 training images, 50 validation images, and 50 test images. All images are downscaled to a resolution of 64x64 pixels, which is significantly smaller than the typical ImageNet resolutions. This reduction not only demands less computational power but also introduces unique challenges.

Despite its reduced size, Tiny ImageNet poses considerable challenges due to the lower resolution of the images. This can complicate the task of feature extraction, which is crucial for effective image classification. Additionally, the wide variety of classes within the dataset encapsulates a broad spectrum of objects, adding to the complexity of the classification task.

Tiny ImageNet has become an invaluable tool for developing image classification models that need to perform well with limited data and reduced image quality. It serves as an accessible yet rigorous benchmark for testing various machine learning techniques, particularly convolutional neural networks (CNNs). This dataset allows researchers and educators to experiment with and explore the capabilities of different algorithms without the extensive computational requirements of larger datasets.

Advancements in the classification performance on Tiny ImageNet have been driven significantly by innovations such as the Astroformer. This model integrates the strengths of both transformer and convolutional architectures, utilizing a blend of deep learning strategies tailored for low-data regimes. The Astroformer has set new performance benchmark obtaining 92.98%. We will concentrate on employing Deep Convolutional Neural Networks to attain the highest achievable accuracy

## IV. Approaches

The training dataset is structured as a collection of folders, totaling 200, each dedicated to a specific class. Within these folders are 500 images corresponding to their respective classes. Notably, included below are examples of images from the 'lion', 'umbrella', 'koala' and 'candle' categories.

Fig. 1. lion

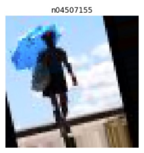


Fig. 2. umbrella

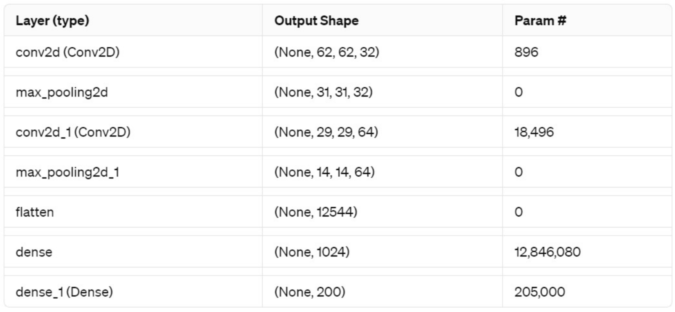
Fig. 3. koala

Fig. 4. candle

Validation data images, on the other hand, are consolidated into a single folder, with annotations provided in a text file detailing class and object locations. To streamline data handling, I developed a Python script to organize these images into separate folders for each class based on the provided annotations. Subsequently, I employed five distinct convolutional neural network models to undertake image classification tasks on the Tiny ImageNet dataset, as outlined below.

### A. Model 1

The proposed Convolutional Neural Network (CNN) architecture is tailored for image classification tasks, employing convolutional and max-pooling layers to extract and process distinctive image features. Notably, regularization techniques, including L2 regularization, are integrated to mitigate overfitting. The model is optimized using the Adam optimizer with a learning rate of 0.0001 and trained via categorical cross-entropy loss minimization over 15 epochs. Evaluation is performed on a separate validation dataset.After 15 epochs, it achieved a training accuracy of 22.96% and a validation accuracy of 14.19%.

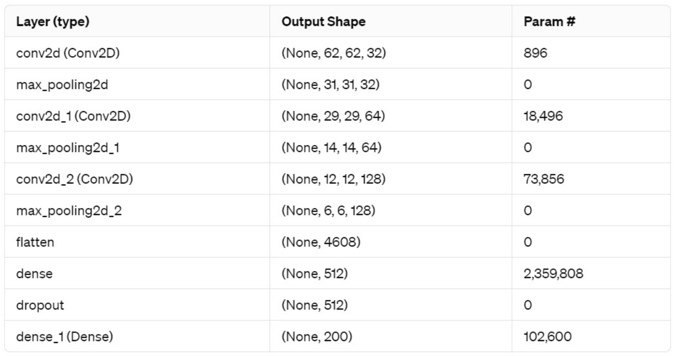


| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 62, 62, 32) | 896 |
| max_pooling2d | (None, 31, 31, 32) | 0 |
| conv2d_1 (Conv2D) | (None, 29, 29, 64) | 18,496 |
| max_pooling2d_1 | (None, 14, 14, 64) | 0 |
| flatten | (None, 12544) | 0 |
| dense | (None, 1024) | 12,846,080 |
| dense_1 (Dense) | (None, 200) | 205,000 |

Fig. 5. Model-1 Summary

### B. Model 2

This CNN model features three convolutional layers with max-pooling for downsampling. The initial Conv2D layer has 32 filters of size (3, 3) and ReLU activation. Max-pooling layers follow each convolutional layer for efficient feature extraction. The final convolutional layer includes 128 filters to capture high-level representations. After flattening, a dense layer with 512 units and ReLU activation is added. Dropout regularization (rate: 0.5) prevents overfitting before the output layer, which employs softmax activation for multi-class classification. Compiled with Adam optimizer and categorical cross-entropy loss, accuracy is used for evaluation. Early stopping and learning rate reduction callbacks are implemented. After 15 epochs, it achieved a training accuracy of 20.17% and a validation accuracy of 22.72%.

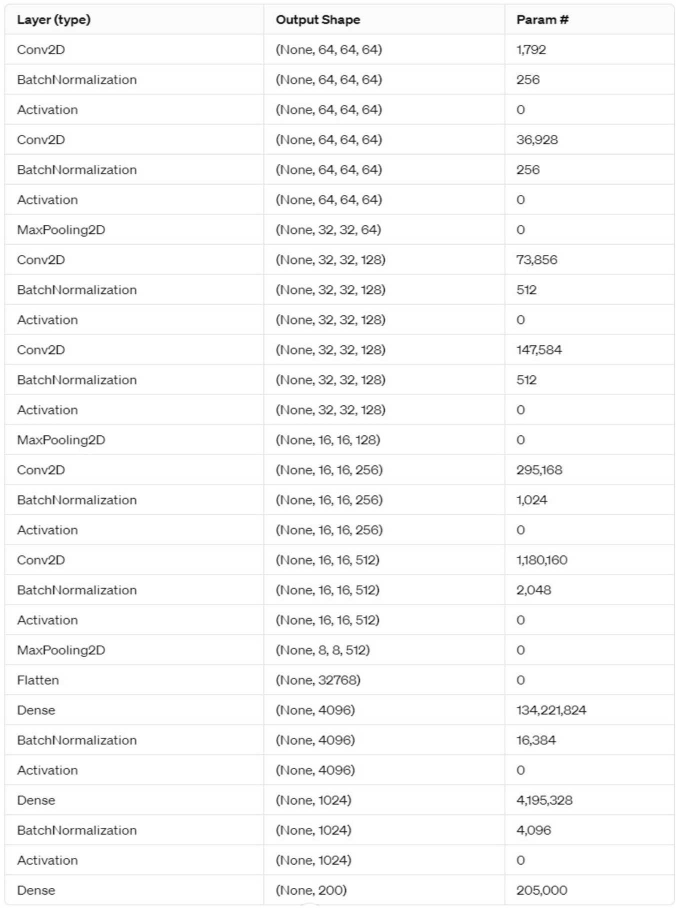


| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 62, 62, 32) | 896 |
| max_pooling2d | (None, 31, 31, 32) | 0 |
| conv2d_1 (Conv2D) | (None, 29, 29, 64) | 18,496 |
| max_pooling2d_1 | (None, 14, 14, 64) | 0 |
| conv2d_2 (Conv2D) | (None, 12, 12, 128) | 73,856 |
| max_pooling2d_2 | (None, 6, 6, 128) | 0 |
| flatten | (None, 4608) | 0 |
| dense | (None, 512) | 2,359,808 |
| dropout | (None, 512) | 0 |
| dense_1 (Dense) | (None, 200) | 102,600 |

Fig. 6. Model-2 Summary

### C. Model 3 - Deep Neural Network - 1

This is the first Deep Neural Network that we developed which is divided into 6 blocks and the model architecture focuses on optimizing feature extraction while maintaining generalization, all without the use of dropout regularization. The initial block, a convolutional layer with 64 filters of size (3, 3) and 'same' padding, initiates the feature extraction process. Batch normalization stabilizes activations, followed by ReLU activation to introduce non-linearity. Subsequent block (2-6) follow a similar structure, each contributing to the network's parameter count without incorporating dropout regularization. The absence of dropout ensures that all learned features are fully utilized, enhancing the model's capacity to capture intricate patterns in the data. The final output layer, a dense layer with softmax activation, transforms raw scores into class probabilities, facilitating multi-class classification. With careful parameterization and the exclusion of dropout, the model maintains a balance between effective feature learning and generalization, making it well-suited for complex image classification tasks.The model underwent training for 15 epochs, resulting in a training accuracy of 29.50% and a validation accuracy of 24.88%. The model summary is provided below

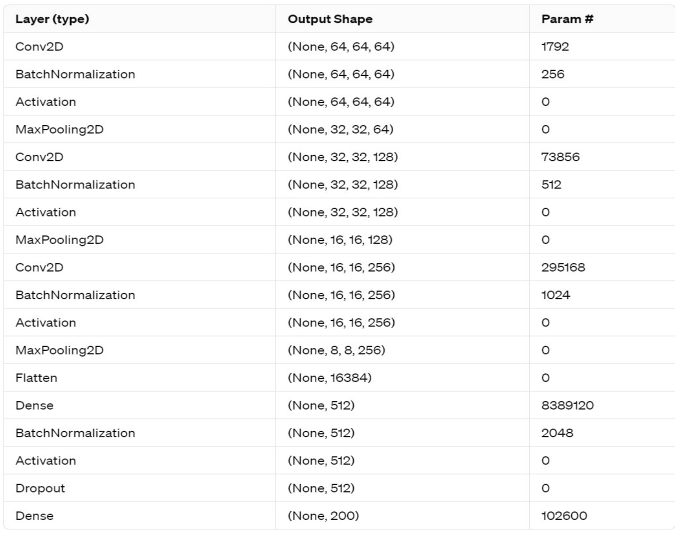


| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2D | (None, 64, 64, 64) | 1,792 |
| BatchNormalization | (None, 64, 64, 64) | 256 |
| Activation | (None, 64, 64, 64) | 0 |
| Conv2D | (None, 64, 64, 64) | 36,928 |
| BatchNormalization | (None, 64, 64, 64) | 256 |
| Activation | (None, 64, 64, 64) | 0 |
| MaxPooling2D | (None, 32, 32, 64) | 0 |
| Conv2D | (None, 32, 32, 128) | 73,856 |
| BatchNormalization | (None, 32, 32, 128) | 512 |
| Activation | (None, 32, 32, 128) | 0 |
| Conv2D | (None, 32, 32, 128) | 147,584 |
| BatchNormalization | (None, 32, 32, 128) | 512 |
| Activation | (None, 32, 32, 128) | 0 |
| MaxPooling2D | (None, 16, 16, 128) | 0 |
| Conv2D | (None, 16, 16, 256) | 295,168 |
| BatchNormalization | (None, 16, 16, 256) | 1,024 |
| Activation | (None, 16, 16, 256) | 0 |
| Conv2D | (None, 16, 16, 512) | 1,180,160 |
| BatchNormalization | (None, 16, 16, 512) | 2,048 |
| Activation | (None, 16, 16, 512) | 0 |
| MaxPooling2D | (None, 8, 8, 512) | 0 |
| Flatten | (None, 32768) | 0 |
| Dense | (None, 4096) | 134,221,824 |
| BatchNormalization | (None, 4096) | 16,384 |
| Activation | (None, 4096) | 0 |
| Dense | (None, 1024) | 4,195,328 |
| BatchNormalization | (None, 1024) | 4,096 |
| Activation | (None, 1024) | 0 |
| Dense | (None, 200) | 205,000 |

Fig. 7. Model-3 Summary

*D. Model 4 - Deep Neural Network- 2*

We designed a deep convolutional neural network (CNN) architecture comprising three convolutional blocks, each contributing to feature extraction. The first block consists of a convolutional layer with 64 filters, followed by batch normalization, ReLU activation, and max-pooling. Similarly, the second and third blocks include convolutional layers with 128 and 256 filters, respectively, followed by the same sequence of operations. After flattening the output, a dense layer with 512 units, batch normalization, ReLU activation, and a dropout rate of 0.5 is added. This architecture facilitates gradual feature extraction with increasing complexity, while dropout regularization helps mitigate overfitting. The model was trained for 15 epochs to achieve a training accuracy of 37.52% and a validation accuracy of 38.61%.



| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2D | (None, 64, 64, 64) | 1792 |
| BatchNormalization | (None, 64, 64, 64) | 256 |
| Activation | (None, 64, 64, 64) | 0 |
| MaxPooling2D | (None, 32, 32, 64) | 0 |
| Conv2D | (None, 32, 32, 128) | 73856 |
| BatchNormalization | (None, 32, 32, 128) | 512 |
| Activation | (None, 32, 32, 128) | 0 |
| MaxPooling2D | (None, 16, 16, 128) | 0 |
| Conv2D | (None, 16, 16, 256) | 295168 |
| BatchNormalization | (None, 16, 16, 256) | 1024 |
| Activation | (None, 16, 16, 256) | 0 |
| MaxPooling2D | (None, 8, 8, 256) | 0 |
| Flatten | (None, 16384) | 0 |
| Dense | (None, 512) | 8389120 |
| BatchNormalization | (None, 512) | 2048 |
| Activation | (None, 512) | 0 |
| Dropout | (None, 512) | 0 |
| Dense | (None, 200) | 102600 |

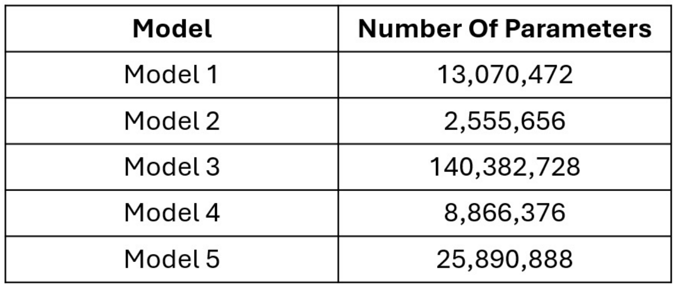
Fig. 8. Model-4 Summary

### *E. Model 5 - ResNet50*

To evaluate the performance of pretrained models we used ResNet50 with their pretrained weights from ImageNet. The training accuracy reached 68.61%, indicating effective learning of the classification task, while the validation accuracy stood at 43.14%, demonstrating satisfactory generalization to new data.

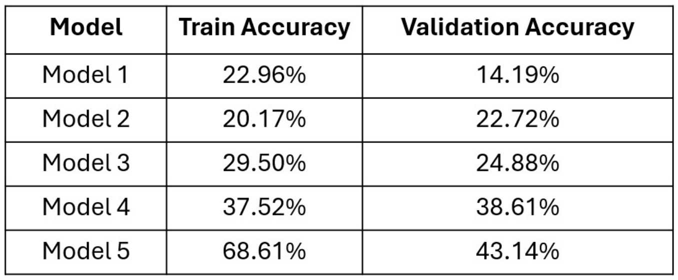
## V. RESULTS

We conducted a rigorous evaluation of five distinct models to classify images within the Tiny ImageNet dataset. Each model underwent iterative refinement, involving adjustments to architectural configurations, regularization strategies, and hyperparameters, to optimize performance while addressing specific challenges inherent to the dataset. Below is a detailed summary of our findings:



| Model | Number Of Parameters |
|---|---|
| Model 1 | 13,070,472 |
| Model 2 | 2,555,656 |
| Model 3 | 140,382,728 |
| Model 4 | 8,866,376 |
| Model 5 | 25,890,888 |

Fig. 9. Number of Parameters



| Model | Train Accuracy | Validation Accuracy |
|---|---|---|
| Model 1 | 22.96% | 14.19% |
| Model 2 | 20.17% | 22.72% |
| Model 3 | 29.50% | 24.88% |
| Model 4 | 37.52% | 38.61% |
| Model 5 | 68.61% | 43.14% |

Fig. 10. Performance metrics

Our experimentation uncovered critical insights into the interplay between model architecture, data preprocessing, and performance outcomes. Given the dataset's inherently low resolution compared to ImageNet, we encountered unique challenges necessitating specialized handling during preprocessing. To mitigate issues associated with low-resolution images, we applied rigorous data augmentation techniques. These included rescaling pixel values, introducing shear, zoom, rotation, and horizontal flip variations, and adjusting brightness levels. Additionally, we carefully managed data leakage by ensuring strict separation between the training and validation datasets throughout preprocessing and model training.

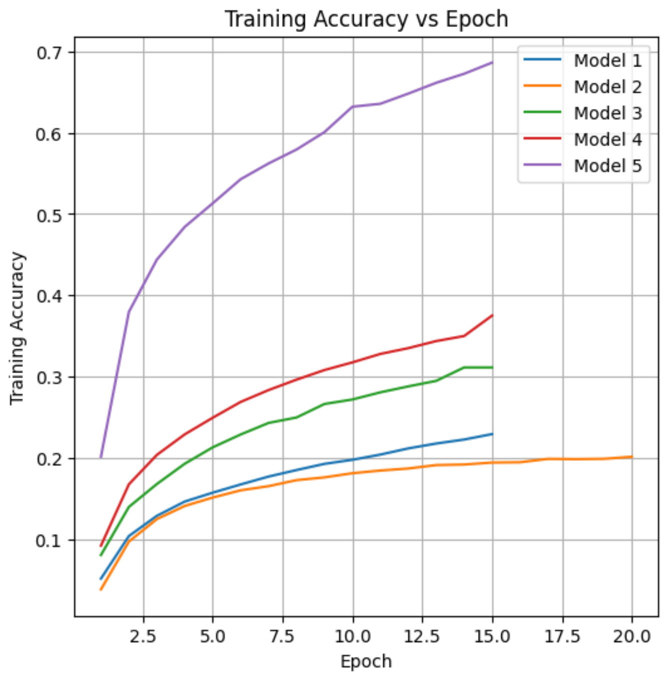


Fig. 11. Training Accuracy vs Epoch

Despite meticulous efforts to address data limitations, our results revealed nuanced interactions between model complexity, overfitting, and validation accuracy. Notably, we observed instances where validation accuracy exceeded training accuracy, indicative of potential overfitting due to model complexity. To mitigate this, we employed dropout regularization techniques and fine-tuned architectural parameters, aiming to strike a delicate balance between model expressiveness and generalization capacity. Through iterative refinement, we successfully navigated these challenges, culminating in Model 5's remarkable performance leveraging a pre-trained ResNet50 backbone.

In Model 5, we capitalized on the power of transfer learning by incorporating a pre-trained ResNet50 architecture, which had undergone extensive training on the vast ImageNet dataset. This approach allowed us to leverage the knowledge and feature representations learned by ResNet50, tailoring them to the nuances of our Tiny ImageNet dataset. By transferring the pre-trained weights and fine-tuning the model's parameters, we achieved significant performance gains. This transfer learning strategy not only expedited the training process but also markedly improved the model's ability to generalize to unseen data. By harnessing the rich feature representations acquired from ImageNet, Model 5 demonstrated superior performance in classifying Tiny ImageNet images, underscoring the efficacy of transfer learning in mitigating data limitations and managing modelcomplexity.

## VI. Conclusions

In this project, we embarked on the task of classifying images from the Tiny ImageNet dataset using various convolutional and deep neural network architectures. We began with a humble 2-layer convolutional network, yielding an accuracy of 14.19%. Progressing to a 3-layer convolutional neural network saw a notable increase in accuracy to 22.72%. However, these simple CNN models struggled to effectively capture the intricate features present in the Tiny ImageNet dataset, characterized by its low resolution and noise levels. Subsequently, we explored deeper neural network architectures, resulting in validation accuracies of 24.88% and 38.61%, outperforming the CNN models. Leveraging the power of transfer learning, we employed a pre-trained ResNet50 model, which attained a validation accuracy of 43.14%, demonstrating its efficacy in handling the challenges posed by the Tiny ImageNet dataset.

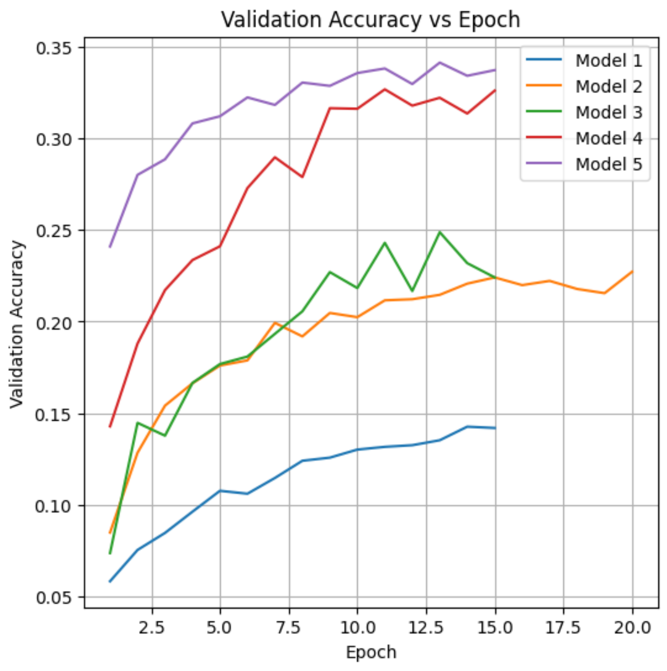


Fig. 12. Validation Accuracy vs Epoch

Achieving high accuracy on the Tiny ImageNet dataset proved challenging due to its low-resolution and scaled-down images. However, by incorporating techniques such as Image Augmentation and Prepocessing, employing Dropout and Regularization, we were able to attain reasonable accuracies. Looking ahead, future endeavors could explore advanced regularization techniques or further fine-tune pre-trained models specifically tailored for the Tiny ImageNet dataset. Additionally, adopting more intricate models like VGG16, Inception,EfficientNet could provide deeper insights and potentially enhance performance by leveraging their extensive parameterization to extract more information from the low-resolution images.

In conclusion, this project underscored the effectiveness of deep convolutional neural networks for image classification tasks on the demanding Tiny ImageNet dataset. Through the evaluation of various deep learning architectures, valuable insights were gained into the most effective strategies for image classification on this dataset. Future work can build upon these findings to refine existing approaches and strive for even higher accuracy levels.

.

## REFERENCES

[1] https://cs231n.stanford.edu/reports/2017/pdfs/931.pdf

[2] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

[3] Dagli, R., 2023. Astroformer: More Data Might not be all you need for Classification. arXiv preprint arXiv:2304.05350.

[4] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[5] Z. Akata, F. Perronnin, Z. Harchaoui and C. Schmid, "Good Practice in Large-Scale Learning for Image Classification," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 3, pp. 507-520, March 2014, doi: 10.1109/TPAMI.2013.146. keywords: Support vector machines;Training;Linear programming;Accuracy;Optimization;Visualization;Encoding;Large scale;fine-grained visual categorization;image classification;ranking;SVM;stochastic learning

[6] `https://www.tensorflow.org/api\_docs/python/tf/keras/preprocessing/image/ImageDataGenerator`

[7] `https://medium.com/@giorgio.martinez1926/nesterov-momentum-explained-with-examples-in-tensorflow-and-pytorch-4673dbf21998`