# Machine Learning Project: Muffin vs. Chihuahua Image Classification Using CNN

## Introduction

This project aims to build a convolutional neural network (CNN) model to classify images of muffins and chihuahuas. The challenge lies in the visual similarity between muffins and chihuahuas, making it a compelling problem for image classification using deep learning techniques. The project employs the TensorFlow Keras framework to implement and train the CNN model.

## Data Preparation

The dataset used for this project consists of 6000 images, divided into two categories: muffins and chihuahuas. The images were sourced from Kaggle's 'Muffin vs. Chihuahua' dataset.

### Dataset Breakdown

- Total images: 6000
- Training and validation set: 4733 images
  - Chihuahuas: 2559 images
  - Muffins: 2174 images
- Test set: Remaining images

The images were preprocessed as follows:

1. Resized to 120x120 pixels to ensure uniformity.
2. Converted to RGB pixel values.
3. Split into training, validation, and test sets.
4. Labels assigned (1 for Muffin, 0 for Chihuahua).
5. Data augmentation techniques applied to enhance model robustness.

## Model Architectures

Three CNN architectures were implemented: AlexNet, VGG16, and ResNet. These architectures were chosen for their proven performance in image classification tasks.

### AlexNet

AlexNet consists of 5 convolutional layers followed by 3 fully connected layers. ReLU activation functions and batch normalization were used after each convolutional layer, and max pooling was applied to reduce the spatial dimensions.

### VGG16

VGG16 employs a simpler structure with only 3x3 convolutions and 2x2 pooling layers throughout the network. This architecture, known for its simplicity and effectiveness, was also implemented and evaluated.

### ResNet

ResNet introduces residual connections to mitigate the vanishing gradient problem, allowing the training of very deep networks. This architecture's performance was compared against AlexNet and VGG16.

## Training Process

The training process involved the following steps:

1. Data Loading and Preprocessing: Images were loaded and preprocessed into tensors suitable for training the neural network.
2. Training Loop: Implemented a loop to pass the input data through the network, calculate the loss using the Binary Cross-Entropy function, and update the model's parameters using backpropagation.
3. Loss Function: Binary Cross-Entropy was used to measure the difference between predicted probabilities and actual labels.
4. Optimizer: The Adam and SGD optimizers were tested with different learning rates (0.005, 0.05, and 0.1).
5. Batch Size: A batch size of 64 was used to train the model.
6. Epochs: The models were trained for 30 epochs.

## Model Evaluation

The models were evaluated using 5-fold cross-validation to estimate performance and prevent overfitting. Performance metrics such as accuracy, zero-one loss, and entropy loss were recorded for each fold.

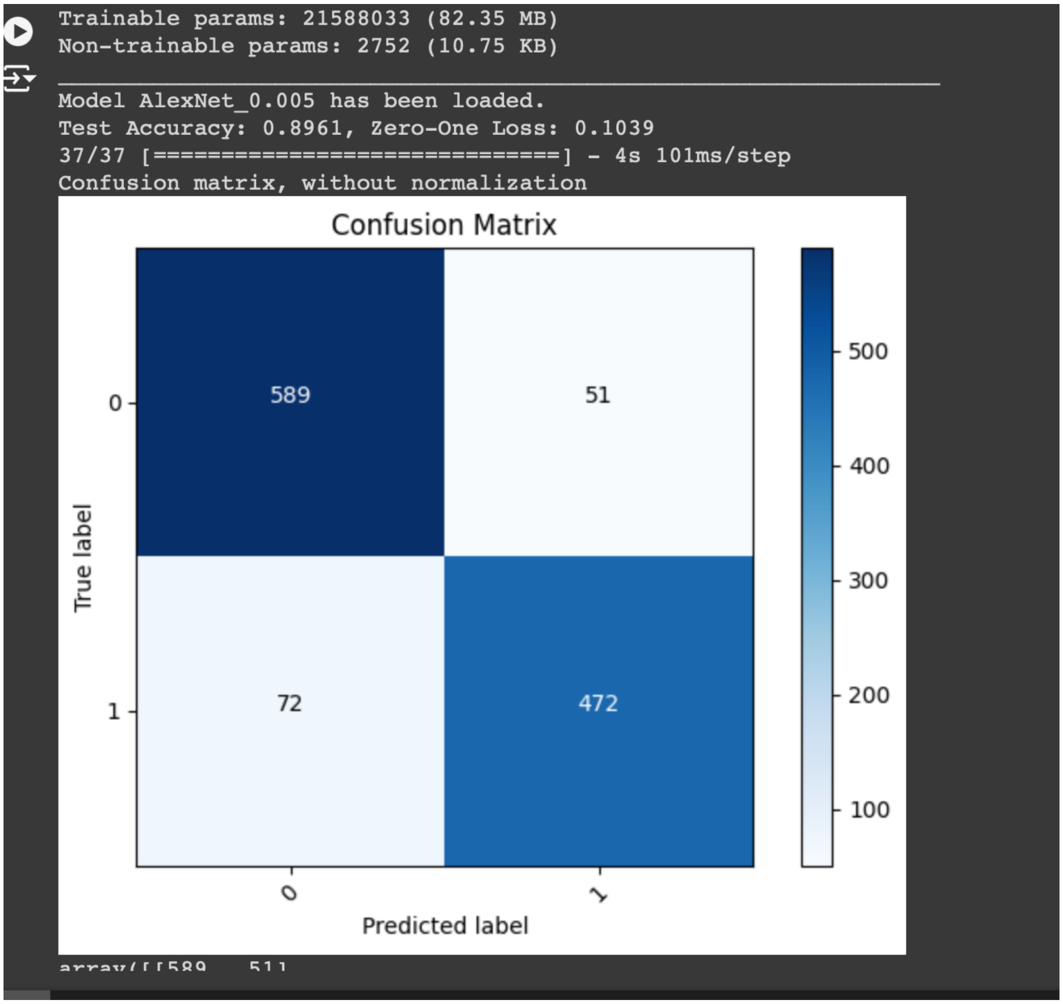
## Experimental Results

The following sections detail the test results for each CNN architecture using the optimal learning rate and optimizer.

### AlexNet

- Optimizer: Adam with a learning rate of 0.005
- Test Accuracy: 0.8961
- Zero-One Loss: 0.1039

Confusion Matrix:

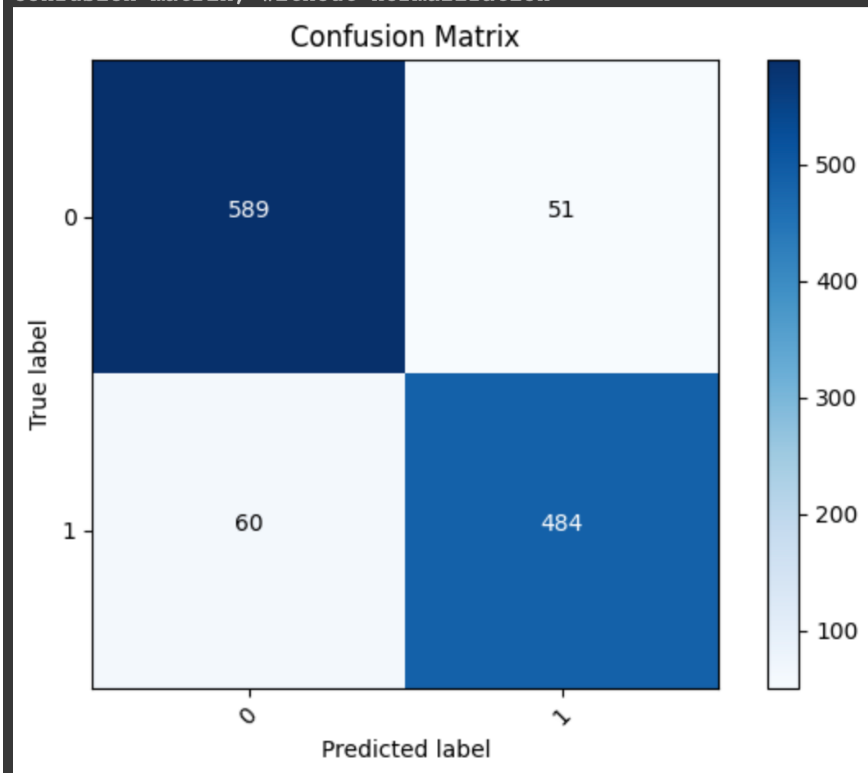



### VGG16

- Optimizer: Adam with a learning rate of 0.005
- Test Accuracy: 0.9062
- Zero-One Loss: 0.0938

Confusion Matrix:

```
Model VGG16_0.005 has been loaded.
Test Accuracy: 0.9062, Zero-One Loss: 0.0938
37/37 [==============================] - 1s 14ms/step
Confusion matrix, without normalization
```



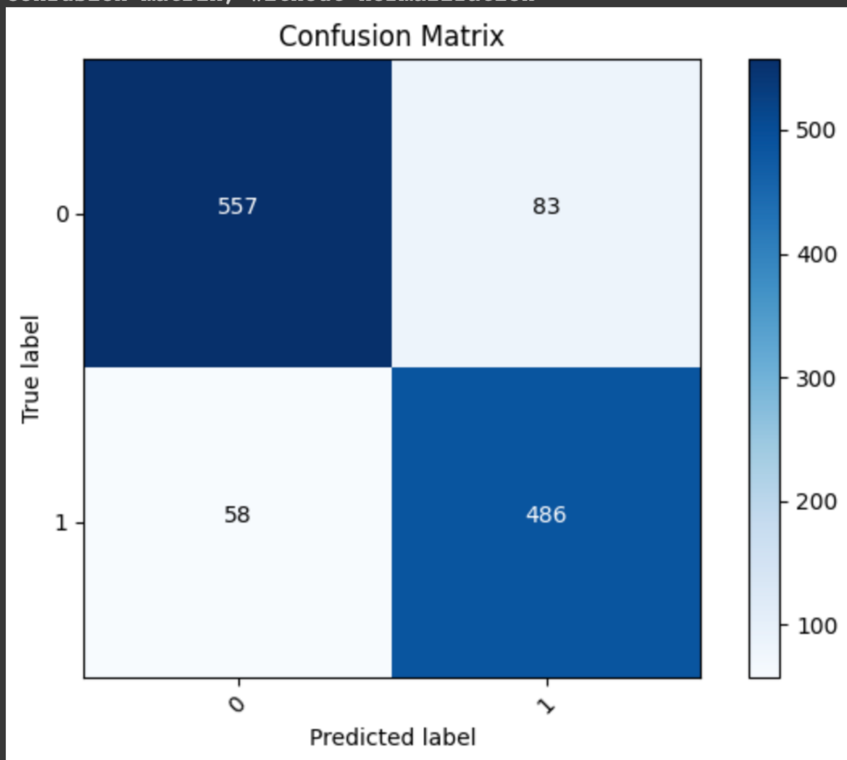

```
array([[589,  51],
       [ 60, 484]])
```

### ResNet

- Optimizer: Adam with a learning rate of 0.005
- Test Accuracy: 0.8809
- Zero-One Loss: 0.1191

Confusion Matrix:

```
Model ResNet_0.005 has been loaded.
Test Accuracy: 0.8809, Zero-One Loss: 0.1191
WARNING:tensorflow:5 out of the last 5 calls to &lt;function Model.make_predict
37/37 [==============================] - 1s 9ms/step
Confusion matrix, without normalization
```




```
array([[557,  83],
       [ 58, 486]])
```

## Summary and Comparison

Comparing the results of the three models:

- Accuracy: VGG16 achieved the highest test accuracy at 90.62%, followed by AlexNet at 89.61%, and ResNet at 88.09%.
- Zero-One Loss: VGG16 also had the lowest zero-one loss at 0.0938, indicating fewer misclassifications compared to AlexNet (0.1039) and ResNet (0.1191).
- Confusion Matrix Analysis:
  - VGG16 showed the best performance in minimizing false negatives (60) and maintaining a high number of true positives (589).
  - AlexNet had a balanced performance but slightly higher false negatives (72) compared to VGG16.
  - ResNet, while still performing well, had the highest number of false positives (83) and the lowest true positives (557).

## Conclusion

The VGG16 architecture, optimized with Adam at a learning rate of 0.005, outperformed both AlexNet and ResNet in classifying muffins and chihuahuas. This project demonstrates the importance of selecting the right model architecture and hyperparameters for achieving optimal performance in image classification tasks.

## References

1. Kaggle Dataset: Muffin vs. Chihuahua Image Classification https://www.kaggle.com/datasets/samuelcortinhas/muffin-vs-chihuahua-image-classification

2. TensorFlow Keras Documentation: TensorFlow Keras (https://www.tensorflow.org/api_docs/python/tf/keras)