Tiny ImageNet Classification using Deep Learning

Presented by,
Narayana Sudheer Vishal Basutkar

Date: April 20, 2023

Outline

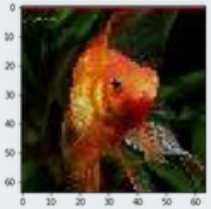
1) Motivation
2) Approach
3) Results
4) Conclusion
5) Future Work

Motivation

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a competition held every year that evaluates algorithms for object detection and image classification on a large-scale dataset of images called ImageNet.

This project focuses on the task of image classification and comparison of six deep convolutional neural networks on the Tiny ImageNet dataset developed using TensorFlow in Python.

Tiny ImageNet Dataset



Imagenet dataset has 1.2 million images and 1000 classes.

Tiny ImageNet is a subset of ImageNet Dataset that has 100,000 images and 200 classes resized to 64X64 pixels with 3 channels (RGB)

Each class has 500 training images and 50 validation images.

Approach

Data Preprocessing and Augmentation

Image pixels were resized between 0 and 1 and were randomly flipped horizontally by setting horizontal_flip to True.

Other Augmentations like zoom and rotation were tested but had very minimal impact mainly due to the small image size of 64X64

Training

Six different Convolution Neural Networks were developed and trained.

All models used the same **ReLU** activations in the hidden layers and **softmax** activation in their output layers.

Adam optimiser with **0.0001 learning rate** was selected after testing different combinations of optimisers and learning rates like RMSProp, SGD and Nadam as it gave the best results.

Model 1 and 2

Model 1

This is the baseline and the simplistic model with just 1 convolution layer that includes 32 filters, 1 max pool layer and 2 Dense layers of 1024 and 200 outputs respectively

Layer (type)	Output Shape
Input	(64, 64, 3)
Conv2D	(None, 62, 62, 32)
MaxPooling2D	(None, 31, 31, 32)
Flatten	(None, 30752)
Dense	(None, 1024)
Dense	(None, 200)

Model 2

This is just one layer more than the first model to show that adding more layers can make the model learn more features and therefore perform better and give a better accuracy.

Layer (type)	**Output Shape**
Input	(64, 64, 3)
conv2d_1 (Conv2D)	(None, 62, 62, 32)
max_pooling2d_1 (MaxPooling2D)	(None, 31, 31, 32)
conv2d_2 (Conv2D)	(None, 29, 29, 64)
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 64)
flatten_1 (Flatten)	(None, 12544)
dense_1 (Dense)	(None, 1024)
dense_2 (Dense)	(None, 200)

Model 3

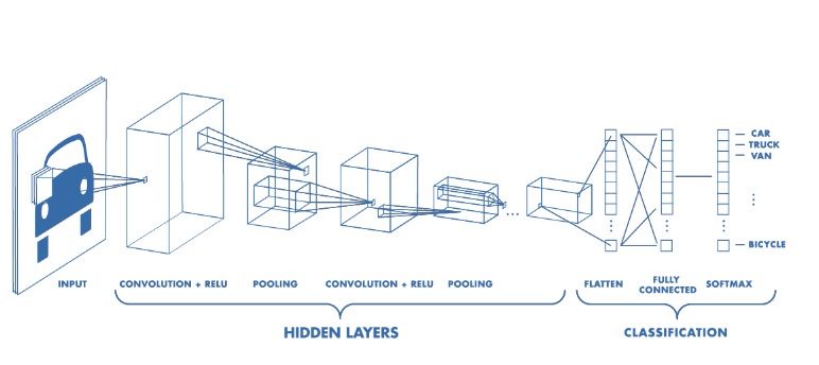


**Image only for example. Does not represent the model.*

Model is divided it into five blocks.

Block 1 -
2 convolutional layers, each with **64** filters. followed by **batch normalization** for each layer, and a **max pooling layer** with a stride of 2X2.

Blocks 2, 3, and 4 -
2 convolutional layers with **128** filters each, followed by **batch normalization** for each layer, and a **max pooling layer** with a stride of 2X2.

Block 5 -
2 convolutional layers with **256** and **512** filters respectively, **batch normalization** for each layer, and a **max pooling layer** with a stride of 2X2.

Upon flattening, **three dense layers** were added with **4096**, **1024**, and **200** outputs, respectively. Batch normalization was applied after the first two dense layers.

Model 4

This model is developed with 5 convolutional layers, max pooling, batch normalization and dropouts with 0.5 rate after each dense layers.

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 64, 64, 64)
conv2d_1 (Conv2D)	(None, 64, 64, 64)
max_pooling2d (MaxPooling2D)	(None, 32, 32, 64)
batch_normalization (BatchNormalization)	(None, 32, 32, 64)
conv2d_2 (Conv2D)	(None, 32, 32, 128)
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 128)
batch_normalization_1 (BatchNormalization)	(None, 16, 16, 128)
conv2d_3 (Conv2D)	(None, 16, 16, 256)
conv2d_4 (Conv2D)	(None, 16, 16, 256)
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 256)
flatten (Flatten)	(None, 16384)
dense (Dense)	(None, 1024)
dropout (Dropout)	(None, 1024)
dense_1 (Dense)	(None, 512)
dropout_1 (Dropout)	(None, 512)
dense_2 (Dense)	(None, 200)

Model 5 and 6

These models are to experiment transfer learning using pre-trained models ResNet50 and ResNet101.

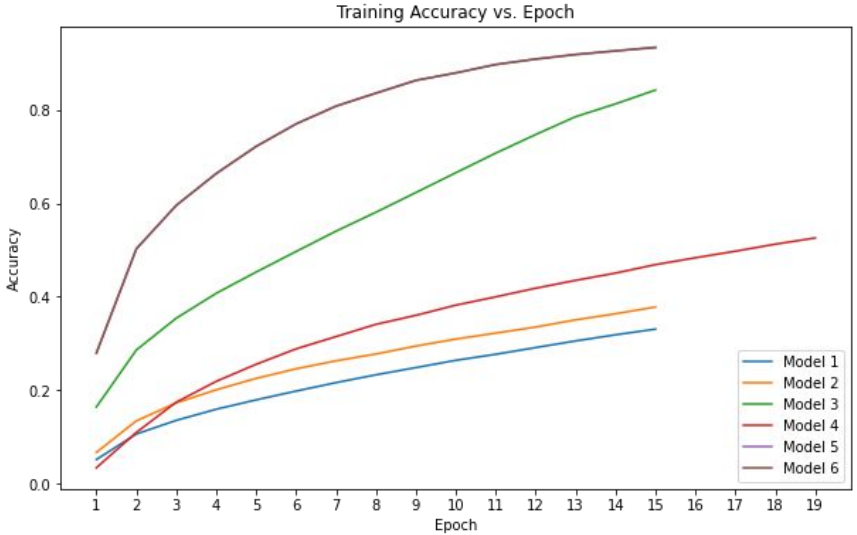
One max pooling layer and one dense layer with 200 outputs were added following the pretrained models

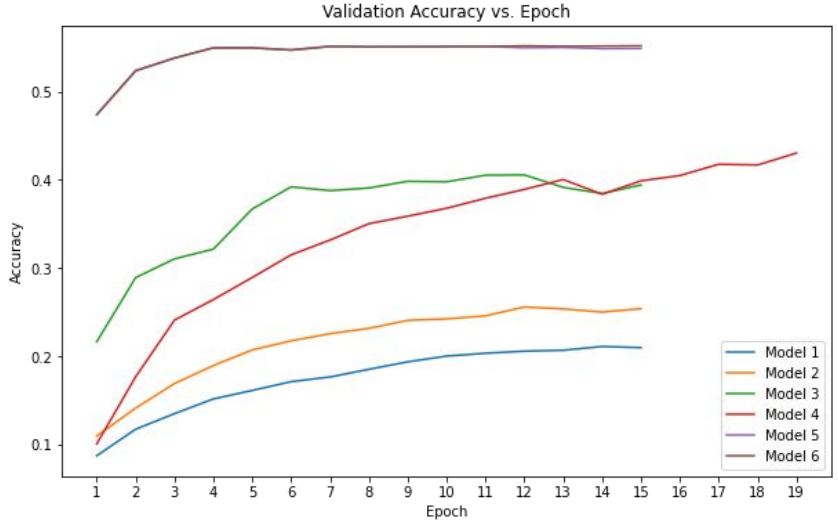
Result Evaluation

Model	Training Accuracy	Validation Accuracy
1	40.84%	22.07%
2	34.94%	25.40%
3	84.26%	39.54%
4	60.89%	42.40%
5	85.86%	55.13%
6	87.94%	55.12%

Model	Number of Parameters	Training Time
1	31,696,968	9m 12s 100ms
2	13,070,472	7m 58s 810ms
3	15,145,992	50m 54s
4	18,404,232	37m 26s 420ms
5	25,890,888	55m 20s 50ms
6	44,961,352	1h 31m 3s 400ms

Result Evaluation





Conclusion

- ➔ Downsampling of the images of ImageNet led to potential problems and hence great accuracies weren't achieved.
- ➔ Upon doing model comparisons, it could be inferred that deep models learn more features and perform better, but are also prone to vanishing gradient problem and overfitting.

Future Work

- ➔ The future direction of this project involves the development of deeper models, such as VGGnet, with carefully chosen regularization techniques.
- ➔ The recent method of using conditional diffusion can be studied and expanded to achieve better results. The model can also be tested on the ImageNet dataset if given the required computation freedom.

Questions?