Assignment 2 Imagenet Report

Group 2: Vignesh Murali, Ritvik Reddy, Ziqing Lu

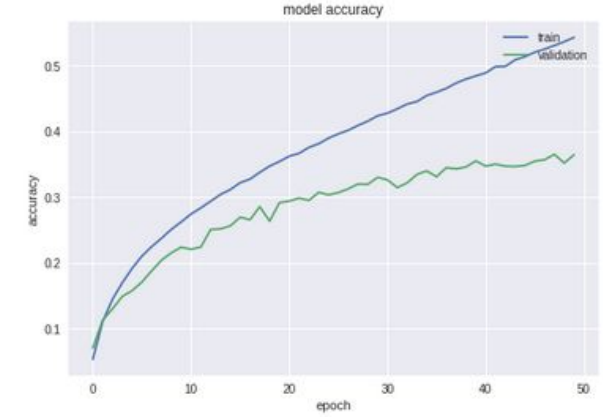
Abstract

This report covers 6 image classification experiments based on tiny ImageNet data (100,000 pictures and 200 classes) and Cifar10 data(60000 32x32 colour images in 10 classes).

Experiment Report

- **Experiment 1: Build best network using Keras**

1. In this experiment we attempt to build the best possible neural network model for the Tiny Imagenet Dataset using Keras framework.
2. Tiny Imagenet has 200 classes. Each class has 500 training images, 50 validation images, and 50 test images. The training and validation sets with images and annotations have been released. Both class labels and bounding boxes as annotations have been provided.
3. CNNs are built using keras and parameters such as depth of network, optimization algorithms, learning rate and epochs are altered.



- **Experiment 2: Use Autokeras to tune hyperparameters**[1]

1. What is Autokeras?
    a. Auto-Keras is an open source software library for automated machine learning (AutoML). Auto-Keras provides functions to automatically search for architecture and hyperparameters of deep learning models.
    b. Key concept behind Autokeras is to explore the search space via morphing the network architectures guided by an efficient Bayesian optimization algorithm

2. How to set up Autokeras:
    a. environment: python 3.6
    b. pip install autokeras

3. Steps to complete a search for mini ImageNet CNN:
    a. organize data for training and testing respectively as required by Autokeras ImageClassifier:
        i. csv file to store the image file name and its corresponding class label
        ii. all image file under one single folder
    b. set the intermediate result path and the searching team and start searching
    c. final fit the train and test data in the best model according to search results

- **Experiment 3:**

AlexNet:

1) AlexNet is a Convolutional Neural Network that rose to prominence when it won the Imagenet Large Scale Visual Recognition Challenge (ILSVRC), which is an annual challenge that evaluates algorithms for object detection and image classification at large scale

2) The ILSVRC evaluates the success of image classification solutions by using two important metrics, the top-5 and the top- 1 error
The Top-1 Error is the percentage of the time the classifier did not give the correct class the highest score while the top-5 error is the percentage of the time that the classifier did not include the correct class among its top 5 guesses.

3) AlexNet received a top-5 error around 16% which was an extremely good result back in 2012. To put in context, the next best result trailed far behind (26.2%). When the dust settled deep learning became cool again and in the next few years, multiple teams would build CNN architectures that would beat human level accuracy

---

[1] https://autokeras.com/
https://medium.com/@santiagof/auto-is-the-new-black-google-automl-microsoft-automated-ml-autokeras-and-auto-sklearn-80d1d3c3005c
https://www.simonwenkel.com/2018/08/29/introduction-to-autokeras.html
https://arxiv.org/abs/1806.10282

Image pre-processing
All the images were scaled to 256×256 pixels.

Data Augmentation:

1)Translation and reflection. This consists of generating image translations and horizontal reflections.

2)The other way they augmented the dataset involved perturbing the R, G, B values of each input image by a scaled version of the principal components

Architecture
AlexNet is made up of eight trainable layers, five convolution layers and three fully connected layers. All of the trainable layers are followed by a ReLu activation.
The architecture also consists of non-trainable layers: Three pooling layers, two normalization layers and one dropout layer (for reducing overfitting).

- **Experiment4:**

VGG-16:

1) Simoyan and Zisserman investigated the effect of neural network depth on its accuracy.They increased the depth of their architecture to 16 and 19 with very small (3*3) convolution layers
2) It got named as VGG (Visual Geometry group) with 16 and 19 layers.

Architecture:

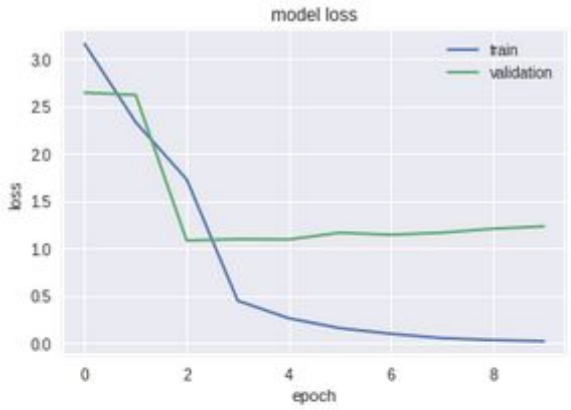
1) The input is basically a 224*224*3 image which passes through first and second layers with 64 feature maps and each having size of 3*3 layers and pooling of stride 14.

2) Next, there are two convolutional layers with 128 feature maps having size 3×3 and a stride of 1. Then there is again a maximum pooling layer with filter size 3×3 and a stride of 2

3)Fifth and sixth layers are convolution layers with filter size 3*3 and stride of 1. Both use 256 feature maps. these conv layers are followed by max-pooling with a stride of 2.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

- **Experiment 5: Transfer Learning (VGG 16)**
  Fine tuning is a process to take a network model that has already been trained for a given task, and make it perform a second similar task.
  i. Assuming the original task is similar to the new task, using a network that has already been designed & trained allows us to take advantage of the feature extraction that happens in the front layers of the network without developing that feature extraction network from scratch.
  ii. It is currently very popular in the field of Deep Learning because it enables you to train Deep Neural Networks with comparatively little data. This is very useful since most real-world problems typically do not have millions of labeled data points to train such complex models.
  iii. Transfer learning works only when the features are general. This means that the features need to be suitable for the bask task and the target task.

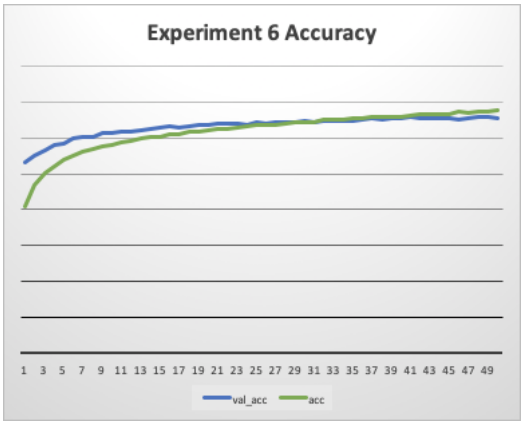


- **Experiment 6: Fine tuning (VGG 16)**[2]

1. What is Fine tuning?

Fine tuning is a method to further adjust the model based on transfer learning, it is an option to realize transfer learning.

2. Steps to Fine tuning:
    a. truncate the last layer
    b. use a small learning rate
    c. freeze the weights of the first few layers
    d. retrain the newly built layers



[2] https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html

Experiment Summary

Exp #	DataBase	Description	Accuracy (train)	Accuracy (val)
1	ImageNet32	Best keras model	0.5432	0.3647
2	ImageNet32	Autokeras	0.55381	0.3744
3	ImageNet32	Alexnet	0.3188	0.2507
4	ImageNet32	VGG 16	0.8681	0.33
5	Cifar10	Transfer Learning VGG16	0.999	0.7034
6	Cifar10	Fine Tuning VGG16	0.6755	0.6559

1. **Transfer Learning is efficient**

   Across all experiments, transfer learning & fine tuning for Cifar 10 delivered the best performance with an accuracy level above 70% and 65% (experiment 5 and 6). And according to our observation, as we don't need to retrain most weights, transfer learning allows a fast progress.

   In experiments 5, we used almost all weights from VGG16, while it got better accuracy than experiment 6, where we re-trained the weights for the last couple of layers. We think it is caused by insufficient data points for a classification problem that has 200 classes to be classified.

2. **Build By Ourselves or Auto Machine Learning?**

   For ImageNet32 training, Autokeras delivered the best performance, a little better than experiment 1. While it delivered an model architecture with 78 layers and took more than 3 days to train.

   As Autokeras needs to explore all potential promising space, it needs time and strong computation power to support its searching. If we do have enough time and GPU and lack enough neural network knowledge for certain problem, Autokeras is a good choice.

	Autokeras	By Hand
Searching Process	Easy, Autokeras will do everything, and balance the	Difficult to reach many promising parameter

	exploration and exploitation	settings
If converge	If enough time, will converge to the best result	Does not guarantee a converge
Network Complexity	Tends to create complex architecture	We always at leat starts from an easy architecture