ASSIGNMENT 2 REPORT
INFO 7374 - Team 11

Dhanashri Palodkar Parag Bhingarkar Rajeshree Kale

The ImageNet project is a large visual database designed for use in visual object recognition software research. More than 14 million images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided. ImageNet contains more than 20,000 categories with a typical category, such as "balloon" or "strawberry", consisting of several hundred images.

Experiment 1

The objective of the first experiment is to design a neural network to train a model for image classification. The dataset used for the experiment is ImageNet. The ImageNet dataset is initially of size 224x224 for the experiment the size of the dataset was reduced to 32x32. The dataset was reduced from 1000 classes to 200 classes. The model was designed with multiple convolutional layers and 2 fully connected layers. From multiple experiments, we observed that to get a better model large kernel sizes were giving better models since the information in images is spread more globally. To train the models better more epochs and low learning rate is preferred.

The model summary:
```
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_48 (Conv2D)           (None, 3, 32, 16)         12816     
_________________________________________________________________
dropout_36 (Dropout)            (None, 3, 32, 16)         0      
_________________________________________________________________
conv2d_49 (Conv2D)           (None, 3, 32, 32)         4640      
_________________________________________________________________
conv2d_50 (Conv2D)           (None, 3, 32, 64)         18496     
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 1, 10, 64)            0      
_________________________________________________________________
conv2d_51 (Conv2D)           (None, 1, 10, 64)         65600     
_________________________________________________________________
flatten_12 (Flatten)         (None, 640)                  0      
_________________________________________________________________
dropout_37 (Dropout)         (None, 640)                  0      
_________________________________________________________________
dense_35 (Dense)             (None, 512)               328192    
_________________________________________________________________
dense_36 (Dense)             (None, 512)               262656    
_________________________________________________________________
dropout_38 (Dropout)         (None, 512)                  0      
_________________________________________________________________
dense_37 (Dense)             (None, 200)               102600    
=================================================================
Total params: 795,000
Trainable params: 795,000
Non-trainable params: 0
_________________________________________________________________
```

ASSIGNMENT 2 REPORT
INFO 7374 - Team 11

Dhanashri Palodkar Parag Bhingarkar Rajeshree Kale

Experiment 2 -Autokeras

Autokeras is a automated open source ML tool which help deep-learning researcher mainly for (2D) Image classification. It is the only thing that is implemented so far and it is missing on many things to be implemented which includes 3D Convolutional Neural Networks, 1D Convolutional Neural Networks, RNNs for speech recognition, normal regression / classifiers RNNs for text, Parallel model training and evaluation Proper memory management; current memory problems cause by PyTorch.
In this experiment we worked on tiny imagenet data of 10000 samples which was resized to 32X32 and then ran the model on autokeras. It Auto-Keras uses network morphism to reduce training time in neural architecture search. Further, it uses a Gaussian process (GP) for Bayesian optimization of guiding network morphism. After running 5 model it gave me accuracy 24% and loss of 14%. Autokeras does hypo Parameter tuning itself.

```
Saving model.
+--------------------------------------------------------------------------+
|        Model ID        |          Loss          |      Metric Value      |
+--------------------------------------------------------------------------+
|           0            |   15.962398386001587   |  0.14440000000000003 |
|           1            |   14.571921348571777   |  0.22359999999999997 |
+--------------------------------------------------------------------------+
```

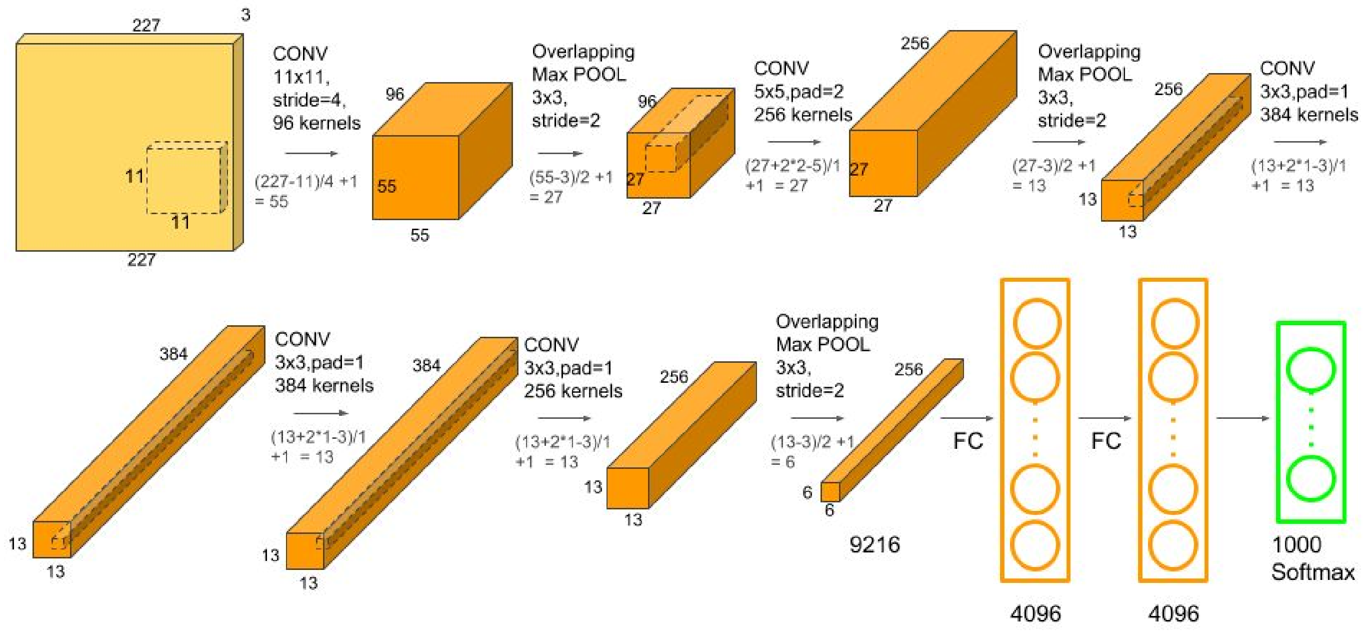
Experiment 3 - Alexnet

For experiment 3 the aim is to train the dataset using AlexNet. AlexNet was the winning entry in ILSVRC 2012. It solves the problem of image classification where the input is an image of one of 1000 different classes (e.g. cats, dogs etc.) and the output is a vector of 1000 numbers. For this experiment, we used only 200 classes instead of 1000. The original AlexNet was designed for images of size 224x224 but we trained the networked for images of size 32x32. The model was performing poorly initially but the after tweaking the model for reduced image sizes the model performed better than before. AlexNet consists of 5 Convolutional Layers and 3 Fully Connected Layers. The total number of parameters for AlexNet is ~800K. The best accuracy for the model was ~12%.

ASSIGNMENT 2 REPORT
INFO 7374 - Team 11

Dhanashri Palodkar Parag Bhingarkar Rajeshree Kale

The Network of AlexNet:



Experiment 4: Densenet201

DenseNet-201 is a convolutional neural network that is trained on more than a million images from the ImageNet database In DenseNet, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. Concatenation is used. Each layer is receiving a “collective knowledge” from all preceding layers.

Increased in the depth of convolutional neural network caused a problem of vanishing information about the input or gradiant when passing through many layers. In order to solve this, authors introduced an architecture with simple connectivity pattern to ensure the maximum flow of information between layers both in forward computation as well as in backward gradiants computation. This network connects all layers in such a way each layer obtains additional inputs from all preceding layers and passes its own feature-maps to all subsequent layers. In Experiment 4 resized imagenet data was trained using densenet on 5 models and the results are as follows:

ASSIGNMENT 2 REPORT
INFO 7374 - Team 11

Dhanashri Palodkar Parag Bhingarkar Rajeshree Kale

Model ID	ImageNet Acc (Top 1)	ImageNet Acc (Top 5)	Params (M)
DenseNet-121	25.02 %	7.71	8.0
DenseNet-169	23.80 %	6.85 %	14.3
DenseNet-201	22.58 %	6.34 %	20.2
DenseNet-161	22.20 %	- %	28.9

The advantages of the denset are:
Decrease the vanishing-gradient problem
Improve feature propogation both in forward as well as backward fashion
Encourage the feature reuse
Reducing the number of parameters

Experiment 5: Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.Transfer learning only works in deep learning if the model features learned from the first task are general.Transfer learning is an optimization, a shortcut to saving time or getting better performance.In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.

BENEFITS OF TRANSFER LEArNING:

- **Higher start**. The initial skill (before refining the model) on the source model is higher than it otherwise would be.
- **Higher slope**. The rate of improvement of skill during training of the source model is steeper than it otherwise would be.
- **Higher asymptote**. The converged skill of the trained model is better than it otherwise would be.

ACCURACY OF THE CIFAR-10 DATASET ON THE BASE MODEL

10000/10000 [==============================] - 1s 79us/step
Accuracy on the Test Images: 0.8200000176429748
So, the CNN model produces an accuracy of 82% on the test dataset. Let's implement transfer learning and check if we can improve the model by using the Densenet201 model, pre-trained on the 'Imagenet weights' to implement transfer learning.

ASSIGNMENT 2 REPORT
INFO 7374 - Team 11

Dhanashri Palodkar Parag Bhingarkar Rajeshree Kale

BUILDING THE MODEL ON THE DENSENET201 FOR THE CIFAR-10 DATASET
DenseNet is a network architecture where each layer is directly connected to every other layer in a feed-forward fashion (within each dense block).DenseNets support any input image size of 32x32 or greater, and are thus suited for CIFAR-10 or CIFAR-100 datasets. There are two types of DenseNets, one suited for smaller images (DenseNet) and one suited for ImageNet called DenseNetImageNet. They are differentiated by the strided convolution and pooling operations prior to the initial dense block. The model build on CIFAR-10 dataset using DenseNet201 network did not yield better performance. Number of epochs used were 10 but the model validation loss and accuracy did not change and hence we need to perform fine tuning of the model to yield better results.
Model Result:
loss: 14.4982 - acc: 0.1005 - val_loss: 14.5385 - val_acc: 0.0980

Based on the above model, by using pre-trained features, the accuracy of the model was intiially 82% on the test data but after applying the DenseNet201 network is was very less. Also, the number of trainable parameters in the transfer model is low as compared to scratch model. Hence the model needs to be further trained more to predict better results.

Experiment 6: Fine Tuning

·The task of fine-tuning a network is to tweak the parameters of an already trained network so that it adapts to the new task at hand. The initial layers learn very general features and as we go higher up the network, the layers tend to learn patterns more specific to the task it is being trained on. Thus, for fine-tuning, we keep the initial layers intact and retrain the later layers of the model.
Guidelines for fine-tuning implementation:
The common practice is to truncate the last layer (softmax layer) of the pre-trained network and replace it with our new softmax layer that are relevant to the problem. For example, pre-trained network on ImageNet comes with a softmax layer with 1000 categories. If our task is a classification on 10 categories, the new softmax layer of the network will be of 10 categories instead of 1000 categories. We then run back propagation on the network to fine-tune the pre-trained weights. Make sure cross validation is performed so that the network will be able to generalize well. Use a smaller learning rate to train the network. Since we expect the pre-trained weights to be quite good already as compared to randomly initialized weights, we do not want to distort them too quickly and too much. A common practice is to make the initial learning rate 10 times smaller than the one used for scratch training.It is also a common practice to freeze the weights of the first few layers of the pre-trained network. This is because the first few layers capture universal features like curves and edges that are also relevant to our new problem. We want to keep those weights intact. Instead, we will get the network to focus on learning dataset-specific features in the subsequent layers.

BUILDING THE SEQUENTIAL MODEL
The model used for transfer learning using the DenseNet201 gave results with less accuracy and loss. So, to make the model more accurate we use fine tuning. Using fine tuning, the model predicts better results and is a better fit.

ASSIGNMENT 2 REPORT
INFO 7374 - Team 11

Dhanashri Palodkar Parag Bhingarkar Rajeshree Kale

BUILDING THE MODEL ON THE DENSENET201 FOR THE CIFAR-10 DATASET

DenseNet is a network architecture where each layer is directly connected to every other layer in a feed-forward fashion (within each dense block).DenseNets support any input image size of 32x32 or greater, and are thus suited for CIFAR-10 or CIFAR-100 datasets.There are two types of DenseNets, one suited for smaller images (DenseNet) and one suited for ImageNet called DenseNetImageNet. They are differentiated by the strided convolution and pooling operations prior to the initial dense block.The model build on CIFAR-10 dataset using DenseNet201 network did not yield better performance. Number of epochs used were 10 but the model validation loss and accuracy did not change and hence we need to perform fine tuning of the model to yield better results.

ACCURACY OF THE CIFAR-10 DATASET ON THE TRANSFER LEARNING MODEL

Test loss: 0.76271309633255
Test accuracy: 0.805

So, the CNN model using DenseNet201 with fine tuning produces an accuracy of 80.50% on the test dataset which is better accuracy and hence yields good results.

Based on the above model, by using pre-trained features, the accuracy of the model jumped from 93% to 80.50% on the test data after applying the DenseNet201 network. Also, the number of trainable parameters in the transfer model is low as compared to scratch model. Apart from this, the CNN scratch model took around 15 minutes to train on CPU, while the transfer model took less than a minute to train the model. We can conclude that the use of Fine Tunning not only improves the performance of the model but also is computationally efficient.