**COP528 AI and Applied Machine Learning Coursework**

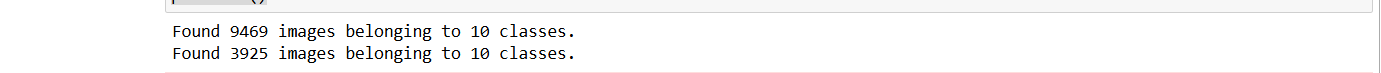
**Introduction for approach selection**

Here I have selected CNN convolution neural network because of its power feature for image processing. It processes data that comes from multiple arrays. For example, a 2D array can be composed of colour images, a 1 D array can be composed of signals and a 3d array can be composed of video. By using CNN, it goes through stages of a process first the input image goes through a convolution layer which uses filters then goes to another convolution layer that extracts the feature layer i.e the output of the first convolution layer to the current layer via a filter bank. A filter bank is a learning weights which is used for feature extraction and the results gets passed down to a activation function called Relu which introduce non linearity which helps the network to learn complex patterns. The feature maps get passed through a layer called pooling layer that reduces the feature maps dimensions by shifting the input by more than one row or column. Since I have a dataset of 10 features of images it feasible to use CNN to classify the data. Since I have a dataset of images, each image having 10 features, it is feasible to use a CNN to classify the data.

**Implementation Details**

Information on Dataset :

The data set is composed of 9,469 images for training and 3,925 images for validation divided into 10 classes of objects.



**Model Architecture**Below is the code that I have used for my model architecture

A screenshot of a computer code

AI-generated content may be incorrect.

Input and convolution layer :- The First convolutional block is the input layer that takes in 64x64 pixels with 3 colour RGB. Then the layer is inputted into Max Pooling function with 2x2 pool size to reduce the spatial dimensions. The same process is taken place with second and third convolutional layer.

Feature processing :- Flatten layer is used to convert 3d features maps into 1d feature vector. So the job of flatten layer is to flatten the 3d feature maps in a single long 1d vector.

Dense layer are fully connected layers that means each neuron is connected to every neurons to the other layer. Here we used dense 512 that displays the capacity to learn complex presentation.

L2 regularization to combat overfitting of the data. This happens when the training data performs too well and performs poorly on unseen data.

Drop out layer is another function which is used to prevent overfitting. During the training of the data it deactivates certain layers in the neuron network thus making the model to learn features without depending on the neurons.

**Training Protocol**

The model was developed by using a learning rate or 0.001 with an optimizer called Adam which is used to update the model weight during training. Loss function is specified the amount of loss occurred during the training of the model. The categorical cross-entropy loss function to quantify the disparity between the predicted and true labels.

**Result analysis**

As we can see wile training the model from epoch 1 to 10 we see an initial accuracy of 0.16 which then increases in epoch 3 with an accuracy of 0.36 this shows that the training data is learning. At the end of epoch 10/10 we get to see a validation accuracy of 0.46 which isn’t a good model to predict with. While view the graph we notice that the training accuracy shows a clear upward trend that shows that the model is learning. When we look at the validation accuracy also shows an upward trend, but it plateaus and starts to decline at the end. This could mean that the model starts to learn but at the end tends to overfit.

A graph with blue and orange lines

AI-generated content may be incorrect.

**Performance improvement strategy**

To improve the performance and accuracy of out model we made sure to add L2 regularization and used geometric transformation to the training and validation dataset. We have also increased the epoch to 50 which further trained the model and gave a better accuracy of 61.9% which is no ideal but was better compared to the first model. The most significant improvements were made by longer training with proper learning rate scheduling, data augmentation applied for improved model resilience, other regularization techniques that enabled training of deeper networks.

Result Analysis after improving the model

A graph of a graph

AI-generated content may be incorrect.

Looking at the graph the training and validation accuracy began around 23 to 30 percent compare to 10 percent for the epoch of 10. The validation accuracy show oscillation compared to training dataset due to smaller size of validation dataset. Once training is completed, both readings are at around 60-65% correct, where training accuracy is higher than validation accuracy.

**Conclusion**

This project demonstrates how CNNs are good for image classification into various classes. We began with good performance, and the addition of data augmentation along with architecture changes gave us significantly improved results. Our best accuracy of 61.9% is significantly higher than random chance (10%) on our 10-class task. Although there is always room for improvement, our solution achieves a reasonable balance between performance and model complexity and makes a good starting point to work from in the future.

References

Yann Lecun, Yoshua Bengio, Geoffrey Hinton. Deep learning. Nature, 2015, 521 (7553), pp.436-444. ff10.1038/nature14539ff. ffhal-04206682