CS549 Project Notes

Validation.py tests and benchmarks various models against each other.

Model 1: A very basic model and ‘tiny’ model.

* Directly flattens image data and feeds each pixel’s input (30 x 30 x 3) to the 43 output nodes. There are thus over 116,000 weights to train this “tiny” model. Performs decently well, averages about 70-80% accuracy with approximately 300 ms/epoch and 500 microsec/step.

Model 2: Small-X model

* Same as model 1 but adds a hidden dense layer of 32 units. Overall performs consistently poor (<10%) compared to model 1. I believe that reducing the 2700 pixel input values down to only 32 neurons results in too much information being lost. Clearly, further models should use larger hidden layers.
* In a dense layer, each neuron in the layer receives input from all neurons of the previous layer. Convolutional layers on the other hand have neurons only receive input from a restricted area of the previous layer. (Probably important to note on the paper)

Model 3: + Convolution, MaxPooling, Flatten layers

* Adds a convolutional layer: applies 32 filters of size 3x3 with ReLU activation directly to the input image. Extracts features from the image by learning various aspects of the image in small receptive fields.
* MaxPooling2D layer: performs down sampling operation that reduces the dimensions of the feature maps. Reduces the computational complexity and overfitting by abstracting the feature maps.
* Flatten layer: converts output from convolutional and pooling layers into a 1D array to be processed by dense layers
* Dense layer: increased to 64 units – better processing the abstracted features extracted by convolution and pooling layers
* Overall much more complex than model 2 due to adding convolution layer which is better for images (abstracting features) whereas model 2 processes raw data. Model 3 also better at capturing hierarchical patterns in the data (especially convolutional and pooling layers which are ideal in capturing spatial hierarchies in images)

Model 4:

* Adds an additional convolutional & pooling layer. First layer increased to 64 filters instead of 32. (higher initial capacity to process input features)
* Batch normalization: Normalizes the activations from the previous layer: stabilizes the learning process by normalizing the input layer by re-centering and re-scaling.
* 2nd convolutional layer: Another set of 64 filters of the same size, increasing the depth of the network to learn more complex features.
* Both maxpool layers: reduce size of features by abstracting feature maps
* Flatten: same as 3
* Dense layer: increased to 128 units
* Output dense layer: always going to be softmax with num categories since that’s how many classifications there are. (This is same for all models)

Model 5: Increasing complexity + dropout (99%+ accuracy)

* Convolutional layer increased to 128 filters of 3x3 (doubled compared 4)
* Increased 2nd convolutional layer too
* Dense layer: doubled also to 256 units
* Dropout layer: dropout randomly sets 30% of input units to 0 at each update during training, helps prevent overfitting. It’s a regularization technique, reduces co-adaptations among neurons, so that neurons become less sensitive to the specific weights of other neurons. Forces network to develop more robust features that are useful in conjunction with different random subsets of the other neurons.
* Each training set also sees a difference sub-network because different subsets of neurons are dropped. This can be seen as training many different neural networks (ensemble), each with slightly different architectures. The final model (which includes all neurons without dropout during testing) can be thought of as an averaging of these smaller, thinned networks
* Increased number of filters and larger dense layer more effectively captures larger datasets (which ours is)

Model 6: (Model becomes too complex)

* Doubled convolutional layers again 128 -> 256
* Double dense layer 256 -> 512
* Increased dropout rate
* Model becomes too complex and accuracy consistently decreased between runs. 99% -> 97%
* Time to train model also more than doubles 5s/epoch -> 11s/epoch
* Complexity of model likely peaked at model 5

Model 7: Same settings as model 5 with 50% dropout rate

* Similar accuracies between model 5 and 7 (99.39% vs 99.38%)
* Only change is dropout rate increased from 30% to 50%
* Model complexity for this use case likely peaked.

Model 8: Same settings as 5 and 7, 40% dropout rate

* Same benchmark metrics as 5 and 7.
* Slightly higher accuracy vs 5 and 7: 99.54% accuracy. Overall best model between performance and accuracy.