Northwestern MSDS-458 Artificial Intelligence and Deep Learning

Assignment 2: Computer Vision

Andrew Stevens

February 7, 2022

(1) Abstract

This paper tests two general types of structures for the purpose of comparing training, performance, and classification accuracy on images of 10 animal or vehicle classes. The two general structures to be tested are dense deep neural networks and convolutional neural networks. The former is a more generic structure which can be applied to a wide variety of problems, while the latter was developed specifically for the purpose of processing visual imagery, or computer vision. The purpose-built structure is a more complex structure and thus requires more time and resources to train but the accuracy results prove its worth.

(2) Introduction

The Canadian Institute For Advanced Research (CIFAR) down-selected and relabeled 6,000 low-resolution images of 10 classes (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks) from a previously common-use dataset (Krizhevsky, 2009). These images will be used to train neural nets in classifying each image into one of the ten classes.

(3) Literature review

The initial methodology used by Krizhevsky focused on using "interesting-looking filters" trained into Restricted Boltzmann Machines and Deep Belief Networks (DBN) for multi-layer models but was not able to achieve useful levels of accuracy. The following year, Krizhevsky (2010) introduced convolutions into his DBN, achieving an error rate of 21.1%. Introducing other steps in the classifier development, such as "AutoAugmentation" (Cubuk et. al, 2019) to automatically test and introduce policies such as translation, rotation, or shearing reduced the loss to 1.5%, improving on the use of the model "ShakeDrop" (Yamada, et al, 2018).

(4) Methods

The model we started with is a fairly simple deep neural network using dense layers with Rectified Linear Unit (ReLU) activation functions, built using keras.

Table 1: DNN With 2 Hidden Layers

```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 784)               2409232
_________________________________________________________________
dense_1 (Dense)              (None, 384)               301440
_________________________________________________________________
dense_2 (Dense)              (None, 10)                3850
=================================================================
```

I then tested two options of structures for a DNN with 3 hidden layers; varying order of the second and third layer.

Table 2: DNN With 3 Hidden Layers

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 784)               2409232
_________________________________________________________________
dense_4 (Dense)              (None, 384)               301440
_________________________________________________________________
dense_5 (Dense)              (None, 512)               197120
_________________________________________________________________
dense_6 (Dense)              (None, 10)                5130
=================================================================
```

In order to test hyperparameters, keras was integreated into scikit-learn’s gridsearch function and tested across the following:

```
n_layers=[2,3]
first_layer_nodes = [512, 384]
activation_funcs = ['sigmoid', 'relu', 'tanh']
loss_funcs = ['binary_crossentropy','hinge',’SparseCategoricalCrossentropy’]
```

A review of the performance metrics across DNN, CNN and EfficientNet training shows how the two-convolution CNN was almost immediately optimized, the DNN were fairly quick to converge, and how EfficientNet overfit the training set and remained.

Due to the fact that many of the most advanced machine learning methods require expensive and complex resources, other research that has been utilized to seek growth in different directions seek to allow less expensive and more common equipment able to achieve, not similar but at least less deficient results. These approaches include studying "model scaling and identify that carefully balancing network depth, width, and resolution can lead to better performance". Tan and Le (2019) developed EfficientNet, trained on ImageNet to be "8.4x smaller and 6.1x faster on inference than the best existing ConvNet".

.

(5) Results

The first set of experiments sought to optimize the structure and parameters of DNNs. Varying order of the second and third layer had no appreciable effect on test accuracy. The highest performance across structures was the model with 3 hidden layers and no regularization, though it took nearly four times as long to train.

Table 3: DNN Results

model type	hidden layers	Regular-ization	training time	train loss	train acc	val loss	val acc	test loss	test acc
dnn	2	yes	31	1.5437	0.476	1.6371	0.4356	1.6054	0.4545
dnn	2	no	38	1.399	0.5029	1.2617	0.5493	1.4066	0.4974
dnn	3	yes	34	1.4633	0.505	1.5861	0.4596	1.553	0.479
dnn	3	no	125	1.1775	0.5811	1.4117	0.4968	1.3988	0.5097

The gridsearch chose a 2-layer DNN with the following other parameters:

```
{'activation_func': 'relu', 'batch_size': 512, 'epochs': 200, 'first_layer_nodes':
384, 'last_layer_nodes': 128, 'loss_func': 'SparseCategoricalCrossentropy',
```

However, given that the resulting test accuracy was lower than some of the initial experimentation results, this may not be the optimal structure, possibly because `patience` was only set to 2 to keep training time reasonable. It did, however, confirm much of the other parameter selection. The hinge loss function is more commonly used for SVMs, and a binary was obviously an incorrect choice given that we have more than 2 classes. An interesting aspect in reviewing the confusion matrices is where the false positives lie. Dogs and Cats are often confused for eachother, as are automobile and truck. These false positives are logical as the features on these objects are similar, and at this low resolution, a human could even make those mistakes. Though perhaps slightly less logical is a ship for an airplane. Review of the gridsearch results in figures 10 and 11 shows how the activation function selection makes a consistently significant difference in performance across the other parameters. Without an appropriate activation function, the neurons do not output the right signal, no matter how the structure is created.

Table 4: CNN and EfficientNet Results

model type	hidden layers	Convo-lutions	reg	training time	train loss	train acc	val loss	val acc	test loss	test acc
cnn	10	2	yes	747	0.5134	0.9707	1.2153	0.7634	1.2415	0.7529
cnn	10	2	no	7066	0.0373	0.9907	1.2403	0.7384	1.2469	0.7395
cnn	13	3	yes	2236	0.3596	0.9238	0.7969	0.7778	0.8231	0.7824
cnn	13	3	no	4998	0.2117	0.9273	0.7538	0.766	0.8231	0.7634
cnn	19	5	yes	1088	0.3606	0.873	0.8203	0.7352	0.8358	0.7387
efficientnet	233	49	yes	1456	0.0307	0.9915	0.9748	0.8322	1.1066	0.8201

The efficiency the discovered EfficientNet structure allowed it to be trained on the CIFAR-10 dataset in 30% less time than the 3-convolution net and reduce error by an additional 6% (higher accuracy would be achieved through improved hyperparameters and more advanced training on

EfficientNet as the authors did, as they were able to achieve 98% accuracy. In our CIFAR-10 EfficientNet model, the biggest source of error still lies, understandably, between the dogs and cats classes. A comparison of Figures 11 and 12 reveals the different purposes of the separate convolutions. Each of the layers recognizes different features within the image, serving to recognize which class is defined by each.

(6) Conclusions. So, what does it all mean?

Preparation, deep search, extensive experimentation and some novel practices are required to achieve high efficiency and accuracy for image classification. In this study, simple dense networks have difficulty discerning similar shapes, and while the CNNs tested achieve significantly better results, the resulting model would not likely be useful in business applications due to high error rates. Although computer vision in general is an incredible accomplishment, CIFAR-10 is a relatively simple problem with only 10 classes. Complex network structures and data preparation must be searched for and tested, and sufficient resources are the path to practicable models.

(7) References

1. Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images (Technical Report). University of Toronto.

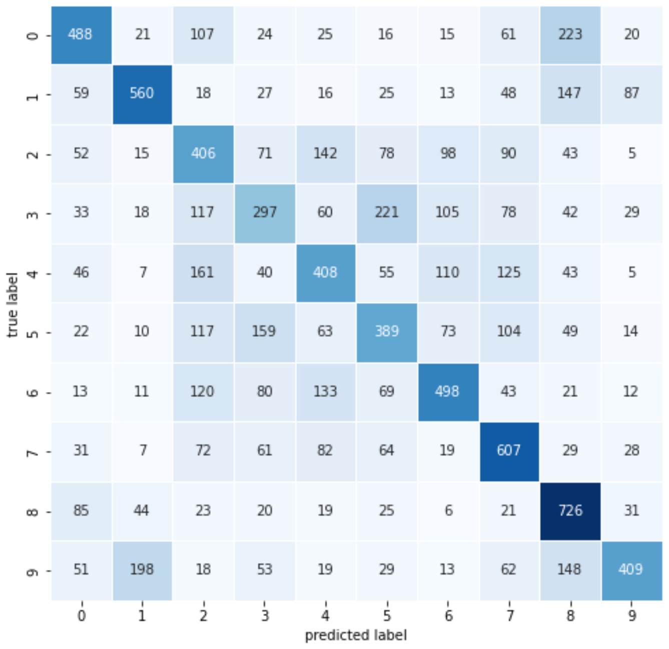
2. A. Krizhevsky (2010). Convolutional deep belief networks on CIFAR-10. (Technical Report). University of Toronto.

3. E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, Q. V. Le, “AutoAugment: Learning Augmentation Policies from Data,” arXiv, 2018.

4. Y. Yamada, M. Iwamura, and K. Kise. Shakedrop regularization. arXiv preprint arXiv:1802.02375, 2018

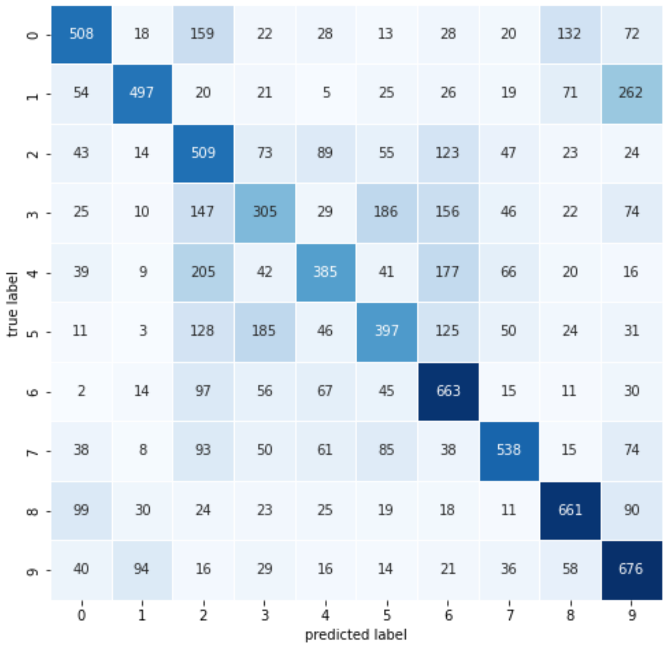
5. M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in International Conference on Machine Learning. PMLR, 2019, pp. 6105–6114

(8) Figures



true label \ predicted label	0	1	2	3	4	5	6	7	8	9
0	488	21	107	24	25	16	15	61	223	20
1	59	560	18	27	16	25	13	48	147	87
2	52	15	406	71	142	78	98	90	43	5
3	33	18	117	297	60	221	105	78	42	29
4	46	7	161	40	408	55	110	125	43	5
5	22	10	117	159	63	389	73	104	49	14
6	13	11	120	80	133	69	498	43	21	12
7	31	7	72	61	82	64	19	607	29	28
8	85	44	23	20	19	25	6	21	726	31
9	51	198	18	53	19	29	13	62	148	409

Figure 1: Confusion Matrix - DNN with 2 hidden layers



true label \ predicted label	0	1	2	3	4	5	6	7	8	9
0	508	18	159	22	28	13	28	20	132	72
1	54	497	20	21	5	25	26	19	71	262
2	43	14	509	73	89	55	123	47	23	24
3	25	10	147	305	29	186	156	46	22	74
4	39	9	205	42	385	41	177	66	20	16
5	11	3	128	185	46	397	125	50	24	31
6	2	14	97	56	67	45	663	15	11	30
7	38	8	93	50	61	85	38	538	15	74
8	99	30	24	23	25	19	18	11	661	90
9	40	94	16	29	16	14	21	36	58	676

Figure 2: Confusion Matrix - DNN with 3 hidden layers

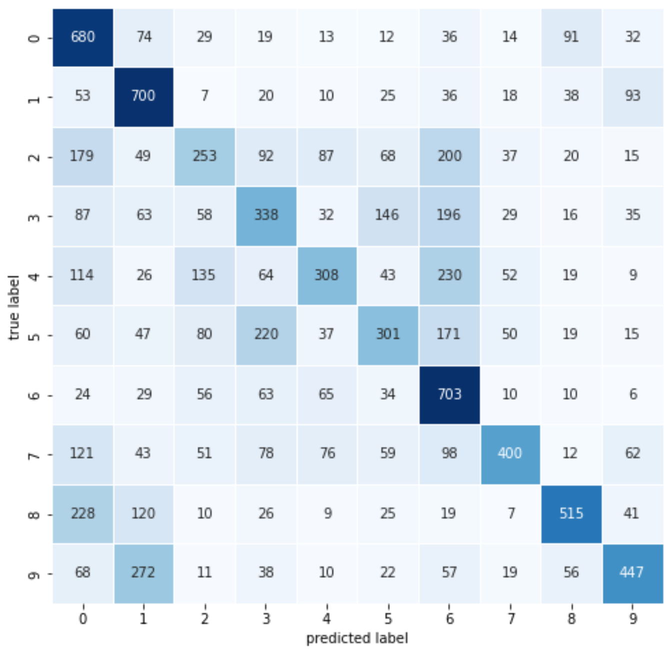


Figure 3: Confusion Matrix – Grid result DNN

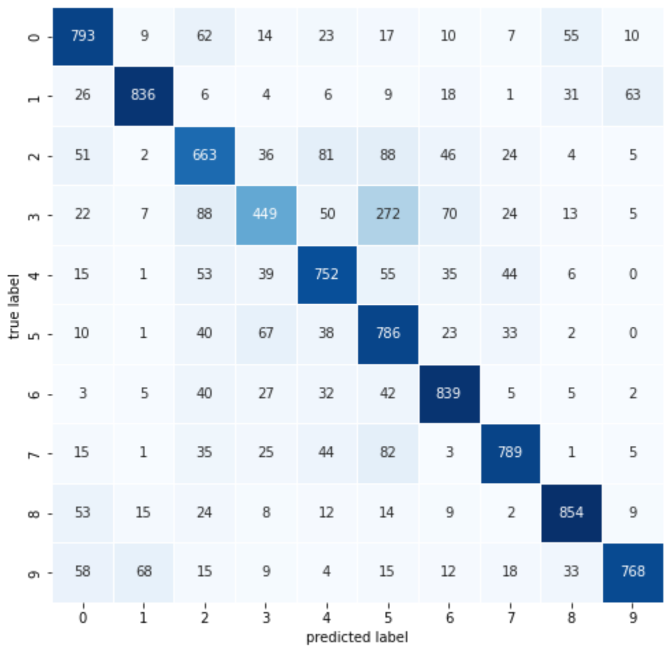
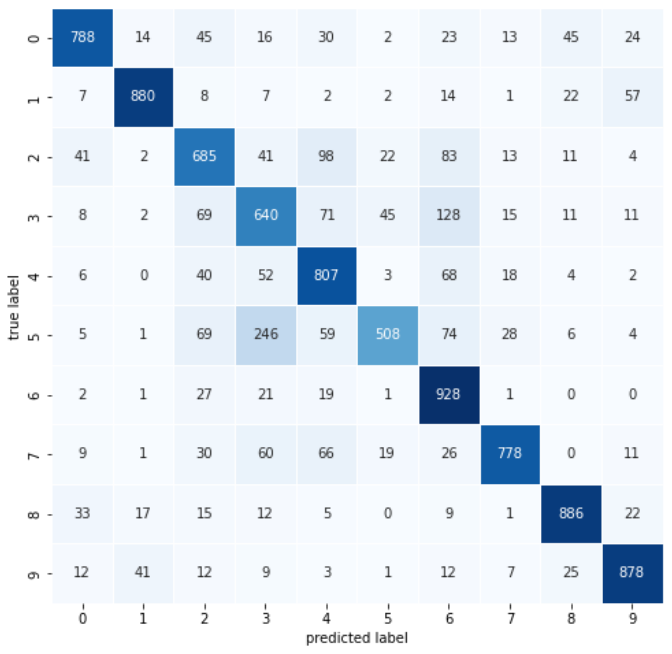
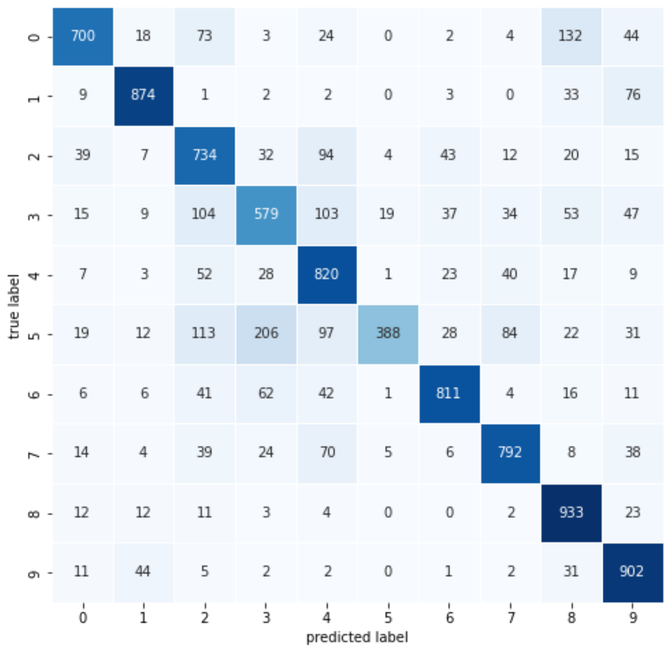


Figure 4: Confusion Matrix - CNN with 2 hidden layers



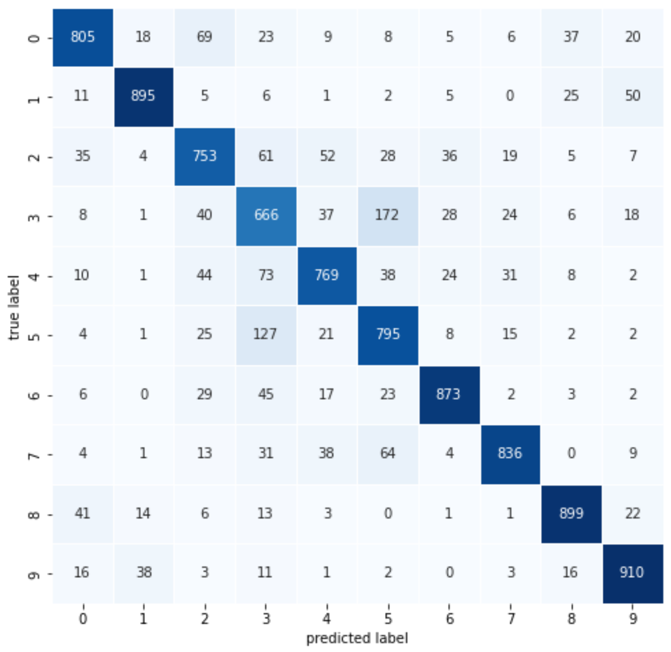
	0	1	2	3	4	5	6	7	8	9
0	788	14	45	16	30	2	23	13	45	24
1	7	880	8	7	2	2	14	1	22	57
2	41	2	685	41	98	22	83	13	11	4
3	8	2	69	640	71	45	128	15	11	11
4	6	0	40	52	807	3	68	18	4	2
5	5	1	69	246	59	508	74	28	6	4
6	2	1	27	21	19	1	928	1	0	0
7	9	1	30	60	66	19	26	778	0	11
8	33	17	15	12	5	0	9	1	886	22
9	12	41	12	9	3	1	12	7	25	878

Figure 5: Confusion Matrix - CNN with 3 hidden layers



	0	1	2	3	4	5	6	7	8	9
0	700	18	73	3	24	0	2	4	132	44
1	9	874	1	2	2	0	3	0	33	76
2	39	7	734	32	94	4	43	12	20	15
3	15	9	104	579	103	19	37	34	53	47
4	7	3	52	28	820	1	23	40	17	9
5	19	12	113	206	97	388	28	84	22	31
6	6	6	41	62	42	1	811	4	16	11
7	14	4	39	24	70	5	6	792	8	38
8	12	12	11	3	4	0	0	2	933	23
9	11	44	5	2	2	0	1	2	31	902

Figure 6: Confusion Matrix - CNN with 5 hidden layers



true label \ predicted label	0	1	2	3	4	5	6	7	8	9
0	805	18	69	23	9	8	5	6	37	20
1	11	895	5	6	1	2	5	0	25	50
2	35	4	753	61	52	28	36	19	5	7
3	8	1	40	666	37	172	28	24	6	18
4	10	1	44	73	769	38	24	31	8	2
5	4	1	25	127	21	795	8	15	2	2
6	6	0	29	45	17	23	873	2	3	2
7	4	1	13	31	38	64	4	836	0	9
8	41	14	6	13	3	0	1	1	899	22
9	16	38	3	11	1	2	0	3	16	910

Figure 7: Confusion Matrix - EfficientNet

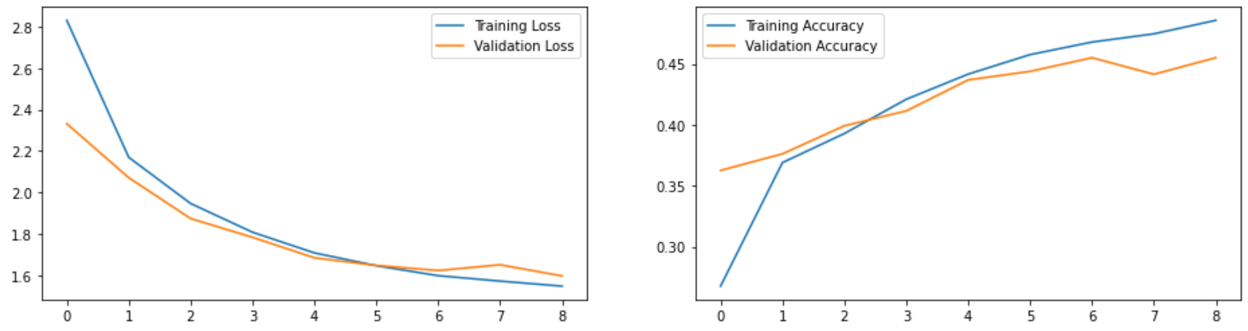


Figure 8: Performance Metrics across Epochs - DNN

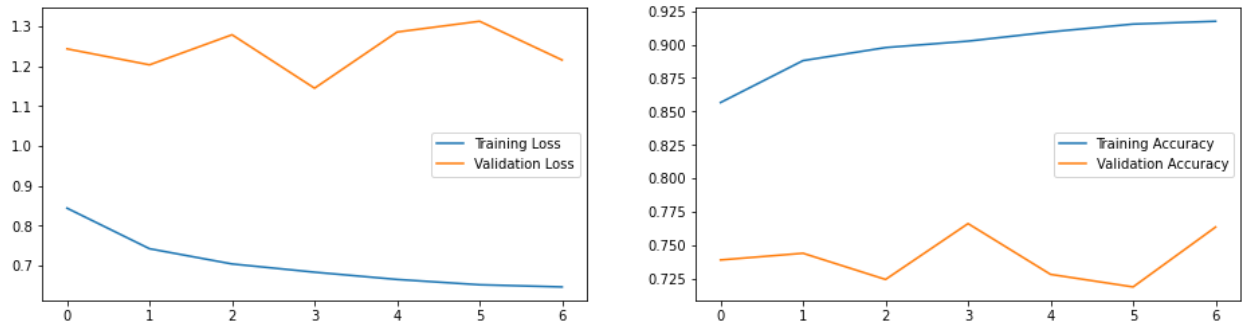


Figure 9: Performance Metrics across Epochs – CNN with 2 convolutions

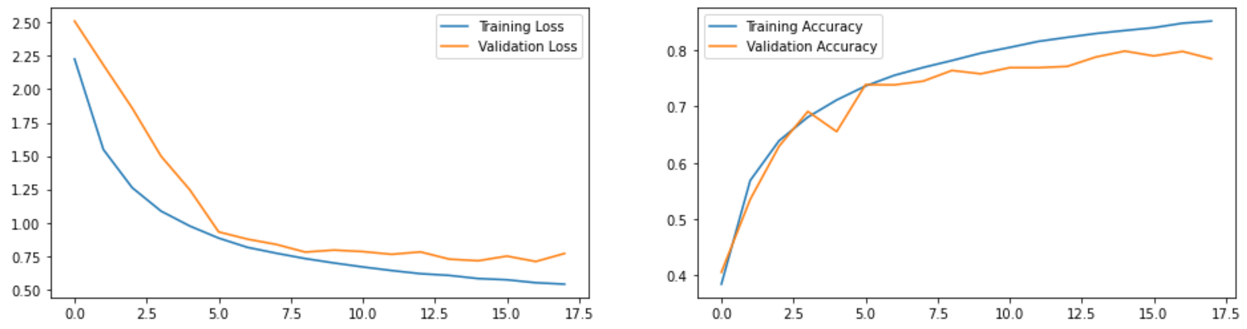


Figure 10: Performance Metrics across Epochs – CNN with 3 convolutions

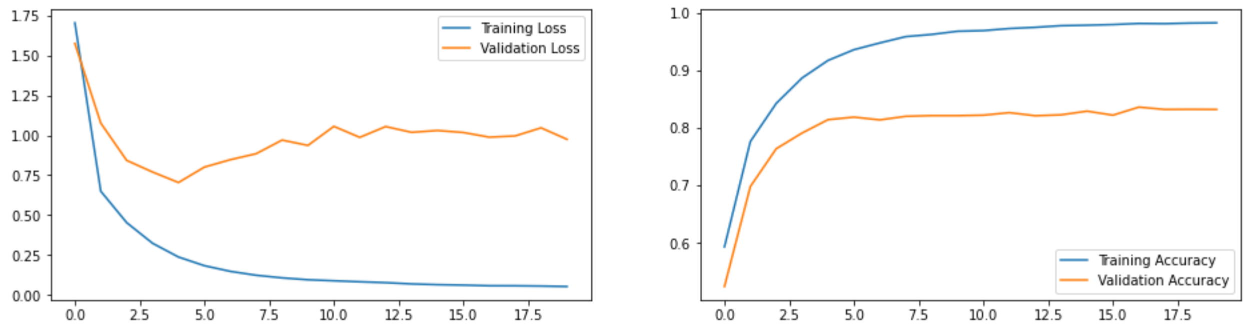


Figure 10: Performance Metrics across Epochs – EfficientNet

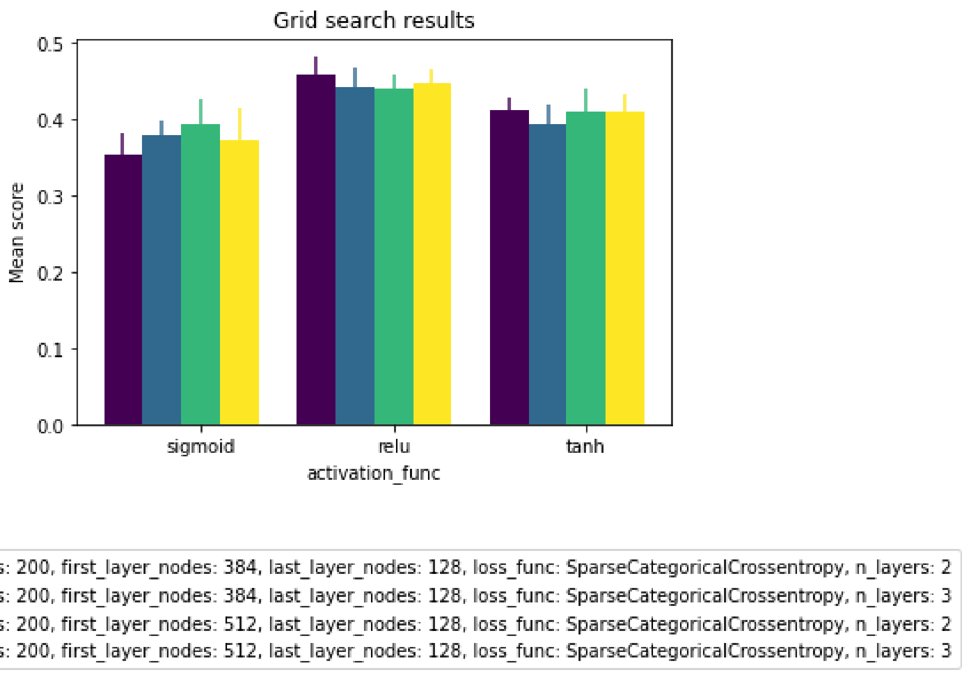


Figure 10: Gridsearch parameter effect - activation

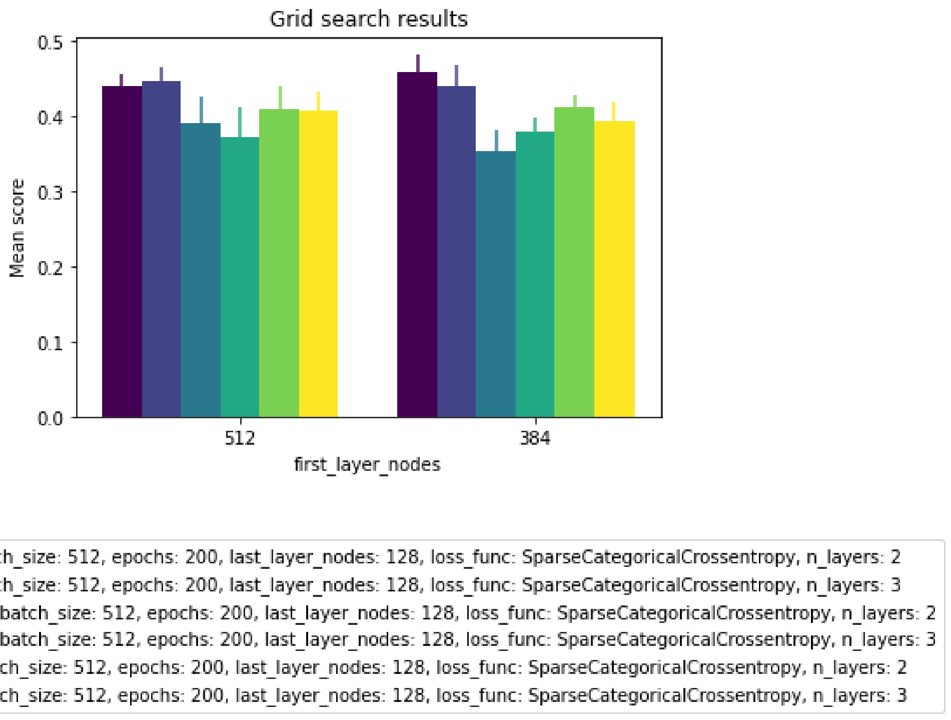


Figure 10: Gridsearch parameter effect – node count

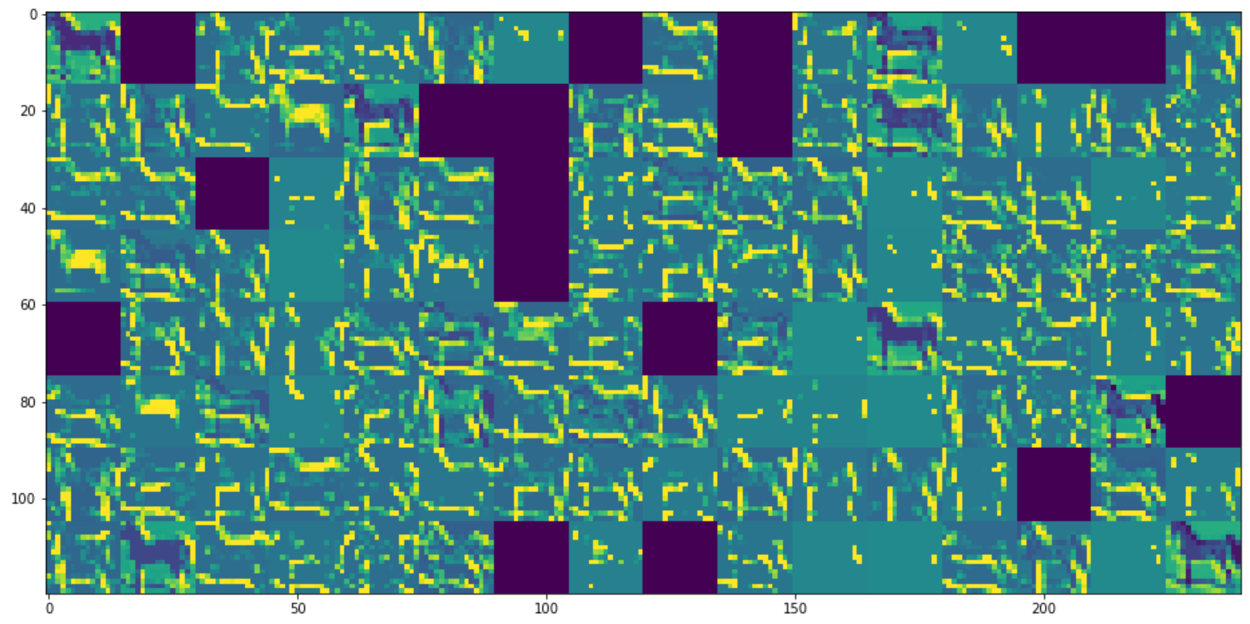


Figure 11: Max pooling 1 activation maps

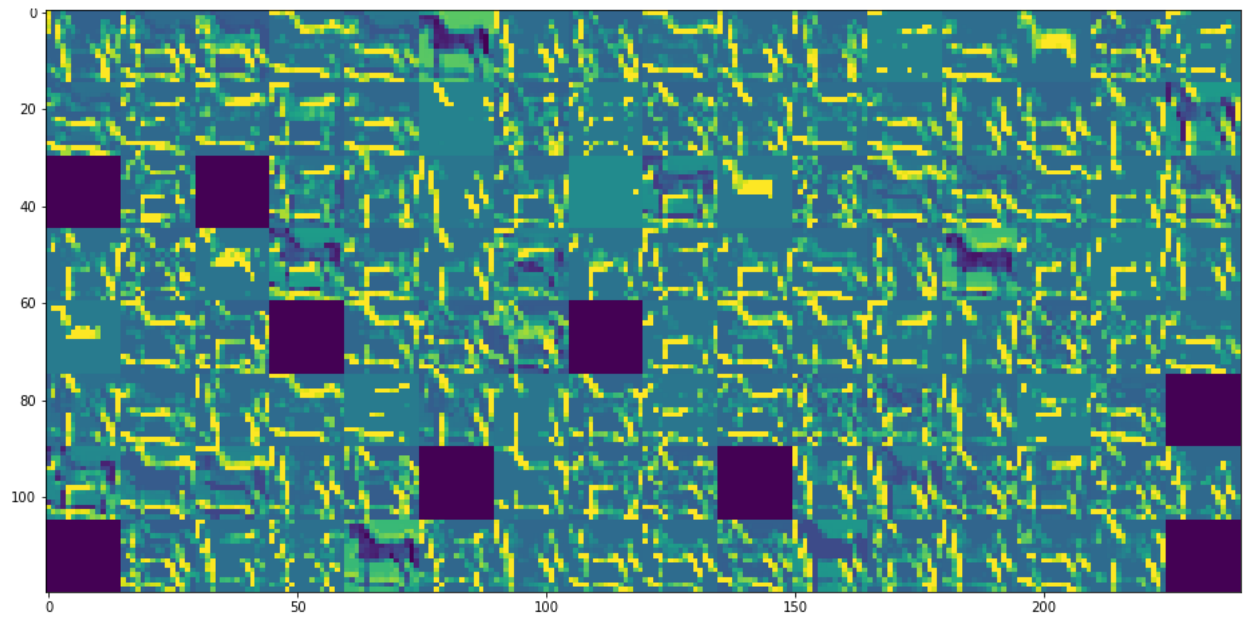


Figure 12: Max pooling 1 activation maps