Network Reduction Technique Based on Neuron Distinctiveness for Classifying Faces-Emotion via Transfer Learning

Samyak Jain
Australian National University
Acton,ACT, Australia
u6734495@anu.edu.au

Abstract. Neural nets with large number neurons and parameters are very common . But , a rule of thumb for obtaining good generalisation is to make sure that we build the smallest system that fits the data. It is essential to prune the network for making it more efficient.The goal of this study was to experiment with the network reduction technique based on distinctiveness. We implemented the technique on a simple feed-forward neural network that dealt with the summarized version of the images and extended our implementation to deep learning that dealt with actual images . Our findings showed us that using deep learning and transfer learning to detect faces-emotion had a higher performance measure than using the summarized version .In addition to this, we experimented with the number of hidden neurons and our research showed us that overall the pruning technique was useful to an extent in removing undesirable and redundant neurons and increasing the performance of our model with a large number of hidden neurons.But on the other hand when the number of hidden neurons are small the pruning had a negative effect on the performance measure.

Keywords: Network Pruning · Faces-Emotion · Deep Learning · AlexNet · Transfer learning ·

1 Introduction

With the rapid advancement in machine learning , much of the progress in the last decade has been a result of deep learning. Deep Learning models these days require a significant amount of computing, memory, and power which becomes a bottleneck in the conditions where we need real-time inference or to run models on edge devices and browsers with limited computational resources. It is therefore the need of the hour to make these networks very efficient. Pruning neural networks helps to obtain good generalisation as it aims to remove undesired parameters in the existing network. The goal of pruning is to reduce the size of the initially made neural network without affecting the accuracy of the network. The results obtained from reducing a large network to a smaller one is better than making a smaller network from the start. Gedeon And Harris[1] provided us with a pruning technique that reduces the number of hidden neurons based on the angle between the pattern vector of the activated output of the hidden layer.

Realistic face data plays a vital role in the research advancement of facial expression analysis systems.All the datasets containing images are always captured in controlled 'lab' environments.The SFEW dataset[2] has been extracted from the temporal AFEW dataset[3] which contains varied head poses, large age range, different face resolutions, occlusions, varied focus and close to real world illumination.These are images which are screen captured from videos corresponding to different movies.First, we discuss how to deal with the The Static Facial Expression Analysis dataset that provides us with features such as Local Phase Quantisation(LPQ) and Pyramid of Histogram Of Oriented Gradients(PHOG) representing the images in a summarized way.To tackle this task we simply made use of a simple feed-forward neural network for image classification.We show how applying the pruning technique affects our feed-forward neural network's predictive performance. Next we extend our research by using deep learning on the actual images of the SFEW dataset.These images are divided into 7 classes namely Anger, Disgust, Fear, Happy, Neutral, Sad and Surprise.For dealing with this task we made use of Convolutional Neural Networks (CNN) and transfer learning .

A Convolutional Neural Network is a type of Artificial Neural network used in image recognition and processing specifically designed to process pixel data.A CNN uses a system much like a multilayer perceptron that has been designed for reduced processing requirements. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers. The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to train but limited for image processing. On the other hand.Transfer learning is a way to transfer the weights of pre trained models and fine tune the model according to our

dataset.Humans have an inherent ability to transfer knowledge across tasks. What we acquire as knowledge while learning about one task, we utilize in the same way to solve related tasks. The more related the tasks, the easier it is for us to transfer, or cross-utilize our knowledge.Similarly, Transfer learning is the idea of overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones. We make use of the pre trained model and just fine tune the model according to our dataset so that it works best for solving our task.

2 Method

2.1 Summary of the Feed-Forward Neural Network and Data Pre-processing:

To begin with we noticed that our dataset had some missing values , so we filled those missing values with the means of their respective features. The class labels were ranging from 1 to 7, but the indexing of classes in a standard neural network starts with zero so we represented our labels ranging from 0 to 6. Then we split our data into training and testing sets with 346 samples out of 675 for training and the rest for testing. This measure was chosen so that it corresponds to the training and testing data used in the original paper of SFEW dataset[2] where they have used 346 samples as training and rest for testing.We scaled our training inputs with a min-max scaler so as to normalise our data from 0-1.Next a simple 3 layer network was constructed with input neurons of the size of our input dimension and hidden neurons set to 20 and outputs neurons to be the size of our output classes. According to Sheela and Deepa[4], a general rule of thumb when defining the size of hidden neurons are given as follows:

- The number of hidden neurons should be 2/3 size of the input layer , plus the output layer.
- The number of hidden neurons should be less than twice the size of the input layer.

Besides this Learning rate of 0.04 with 500 epochs , RELU activation function on the hidden layer and Adam optimiser are used.The concept of Cross Entropy loss was used to calculate the overall loss.[5]

2.2 Data pre-processing for Deep learning

The images in the SFEW dataset were images that are screen captured from running videos of movies.So naturally these images are not the typical images that are used for deep learning. To make these images more useful for our experiments and research we first build a face detector using python's MTCNN package.MTCNN(Multi-task Cascaded Convolutional Neural Networks) is an algorithm that detects coordinates of the faces in an image along with 5 point facial landmarks.Using these coordinates, we cropped the images such that they only contain the faces and not the background.The relevance of this preprocessing step is indicated in Table.1 where we can observe that using face detection improves the model's performance.

Model	Train Accuracy	Test Accuracy
Before Face Detector	86.43%	40.42%
After Face Detector	91.23%	52.94%

Table 1 Average Train and Test Accuracy obtained before using face detection and after using face detection .

In addition to this , we performed data augmentation on the training set with different transforms such as resizing our images to standard 'ImageNet'size (224,244), RandomRotation of images, Random horizontal flipping of images and normalized the images's RGB channel to standard 'ImageNet' mean and standard deviation of (0.485, 0.456, 0.406),(0.229, 0.224, 0.225) .

2.3 Selection of model

Often we use deep learning or transfer learning for processing large datasets.Since our dataset has only 675 images we tried to implement different models to see which type of model was the best fit for our data.First we tried using our own standard CNN model with only 1 convolutional layer and pooling .Then we extended our model with one more convolutional layer and pooling layer.Further we tried using pre trained models such as Resnet18, Resnet50, Resnet-152[6] , Vgg-13[7] and AlexNet[8]. We froze all the layers of these pre trained models except the last fully connected layers and fine tuned our model accordingly.The results from these models are described in Fig. 2.

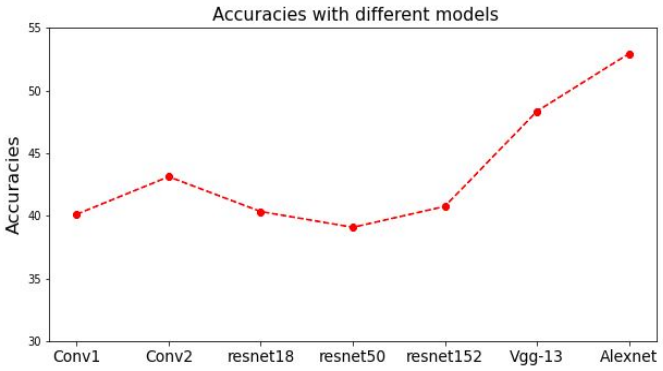

Fig. 1 Average Accuracies obtained on test set with different models. Conv1 refers to CNN with 1 convolutional layer and Conv2 refers to CNN with 2 convolutional layers.

From Fig.1 we can observe that using AlexNet with fine tuned fully connected layers gives us the best performance out of all the models mentioned. Since all other networks such as Resnet50, Resnet152 are very deep models and works best for large amounts of training data which is not suitable in our case. On the other hand Alexnet is not so deep like other models and hence performed better than the rest. So, After observing the results we decided to go ahead with Alexnet as our model for our findings and experiments.

2.4 Hyperparameters Experimentation

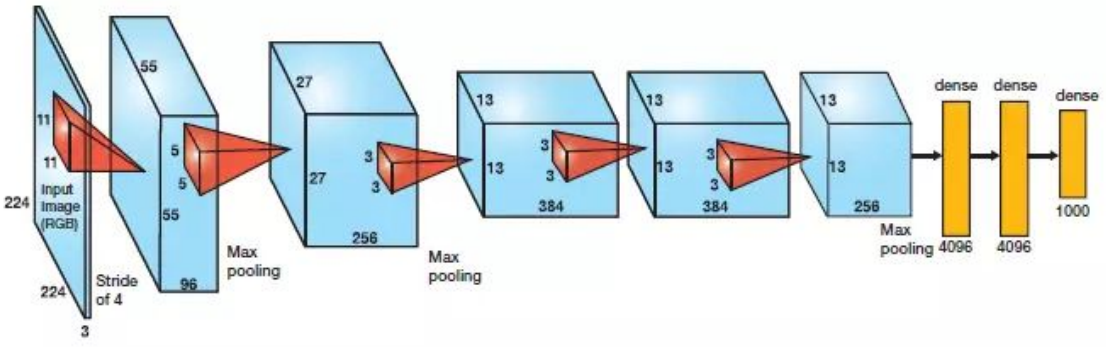

Fig2. Alexnet Architecture. Three dense layers(yellow) in the original model.We used only 2 dense layers.

For Fine tuning the pretrained AlexNet we froze all the layers except the last fully connected layers. Instead of the original 3 fully connected layers we used only 2 fully connected layers. The first layer takes the output of the Average Max Pool convolved layer (9216 features) as input with 200 hidden neurons and instead of the last fully connected layer with 1000 output neurons for 'ImageNet', we used 7 output neurons for classifying emotion.
For optimization of our model , the points for investigation were :

- What training and testing split to use
- Optimal learning rate
- Number of Epochs

From the findings in Table 2 we observed that model with learning rate of 0.001 and 200 hidden neurons with 50 epochs and 80-20 train test split performs best.It is often recommended to give small learning rates when we are trying to fine tune a pre-trained model, so we tried our experiments with small learning rates.

		Learning Rates		
	Epochs	0.001	0.002	0.003
70-30	30	47.22%	51.23%	48.49
	50	55.84%	52.32%	49.08
	80	47.36%	44.67%	45.63
		Learning Rates		
	Epochs	0.001	0.002	0.003
80-20	30	49.06%	51.97%	49.21%
	50	56.84%	53.32%	51.38%
	80	48.92%	46.21%	46.13%

Table 2. Average test accuracies obtained using different training testing split on the left , different learning rates and different number of epochs.

2.5 Finding angle between Hidden Neurons

From our model, we make use of the two fully connected layers and prune the hidden layer output which is the sigmoid activation applied on the hidden layer .This hidden layer output is of the size (input_neurons,hidden neurons) where every column is our hidden pattern vector and we aim to find angle between all pairs of columns. According to Geoden and Harris[1], all these values should be normalised between -0.5 to 0.5 so that the range of angle is 0-180 degrees instead of 0-90.Then , we find the angles between all pairs of columns by using cosine similarity. Gedeon and Harris[1] suggested that vectors between which the angle of separation is below 15 degrees are considered sufficiently similar and one of them is removed . The weight vector of the unit which is removed is added to the weight vector of the unit which remains. Similarly, vectors between which the angle of separation is greater than 165 degrees are considered to be complementary and both should be removed as they are in opposite directions of the output.

Pair of Units	Degree of Vectors(Angle)
6 7	61.29
6 8	155.12
6 9	50.96
6 10	27.72
6 11	30.18
6 12	27.18
6 13	9..18
6 14	29.35
6 15	43.43
6 16	27.18
6 17	27.18
6 18	30.26

Table 3. Pair of unit vectors and the corresponding angle between these vectors. Angles below 15 are similar and angles above 165 are considered complementary.

2.6 Pruning our model

As we have calculated the angles between the unit vectors we now know which neurons we have to prune. Our first step would be to remove all pairs of neurons that are complementary (greater than 165) to each other . This will make our adding of weight vector calculation for similar neurons easier as complementary nodes would be removed first. To do this , first we set the corresponding row of the weight vector of both complementary neurons to zero. Second we take the corresponding column in the output weight vector and set it to zero. And , at last we set the corresponding row of bias vector of the hidden unit to zero. Now since we have deleted the neuron pairs which are complementary , it will ease the computation and addition of the similar neuron vectors. To do this, we first made a clone of all the weight and bias vectors of our original model. After finding out the indices of the similar pair of neurons , the weight vector of the neuron (at first index) is added to the weight vector of another neuron(at second index), and then the weight vector of the former neuron is set to zero. These steps would ensure that the size of the network has reduced from the original one. The last thing to do is to edit the weights and bias of the original model so that now our model works with the new reduced sets of weights and bias. For this we created another model and initialised all the layers of this new model with the layers of our old model. Then we edited the weights and bias of our new model with the new sets of features we got from pruning. This would make sure that our model would not need to be retrained and we can directly test our new model on the training set.

3. Results and Discussion

3.1 Summary of the pruning effect from our Feed-Forward Network

Before pruning the training accuracy we achieved was 47.11% and the test accuracy achieved was 17.63 %. Then after performing pruning and creating a new model with our new reduced network we find that our accuracy on the test set decreases to 14.29 %. From the results obtained it was observable that when the number of neurons pruned are more than 8 (out of 20 that we started initially with) our reduced network performs better than the original model. But when less neurons are pruned (between 0 or 8) the original model performs better on the test set. It is evident that 20 hidden neurons are very large for our network, since half the number of neurons are getting pruned almost every time. So we can observe that whenever the reduction rate of our network was more than 40% the reduced network performs better.

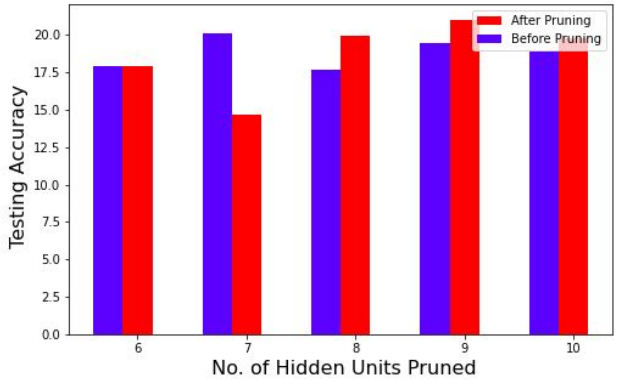

Fig 3. Accuracy before and after pruning with different number of neurons pruned

From Fig.3 we can observe that when large number neurons are getting pruned (greater than 40%) than the model after pruning performs better than the original one, but on the other hand when a small number of neurons are getting pruned (less than 40%) the original model performs better.

3.2 Optimal number of hidden neurons for deep-learning model

In Table 2 all of the settings were also experimented with the number of hidden neurons. The test accuracy achieved on average with different numbers of hidden neurons was 52.94%. But the test accuracy peaked at 200 hidden neurons which gave 56.84% accuracy.So the optimal number of hidden neurons chosen was 200. We experimented with different numbers of hidden neurons and from Fig.4 it is observable that as the model complexity got higher and higher (large number of hidden neurons) the model tends to overfit more and was not able to generalize better.Thus, we see a drop in testing accuracy as we increase the amount of hidden neurons. Since our dataset is small , having less than or equal to 400 neurons seemed optimal for the model rather than the usual large numbers of hidden neurons used for 'ImageNet' to process millions of images.

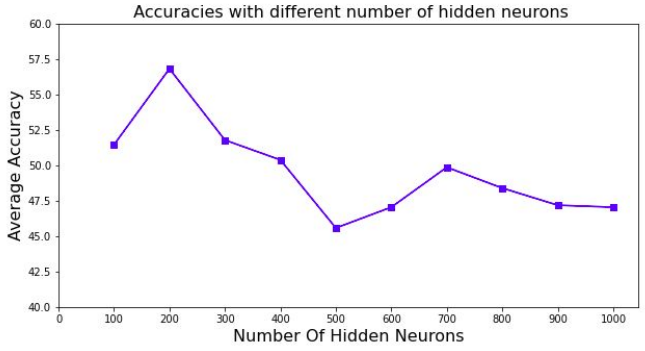

Fig.4 Test Accuracy obtained with different numbers of hidden neurons. Models tended to overfit more as model complexity increased.

3.3 Investigation of the model's reduction rate

After performing pruning, we experimented with our results based on the number of neurons pruned with varying quantities of hidden neurons. For this we define a method that gives us a compression ratio of our model. The Compression rate is the ratio of number of neurons pruned to the total number of initial neurons. From our results it was noticeable that the optimal number of hidden neurons our models reduced to was approximately between 200-300 neurons. As discussed earlier, the model with 200 hidden neurons performed better on the test set than any other model. It is evident from Fig.5 that the model always tried to reduce the number of hidden neurons approximately close to 200-300 neurons. The compression rate and the number of initial neurons seemed to have a positive correlation . As we increased the size of hidden neurons the compression rate got higher and higher . With 200 hidden neurons the compression rate was 26.5% , while with 500 neurons the compression rate went up to 64.5% and with 1000 initial neurons the compression rate spiked up to 69.4%.

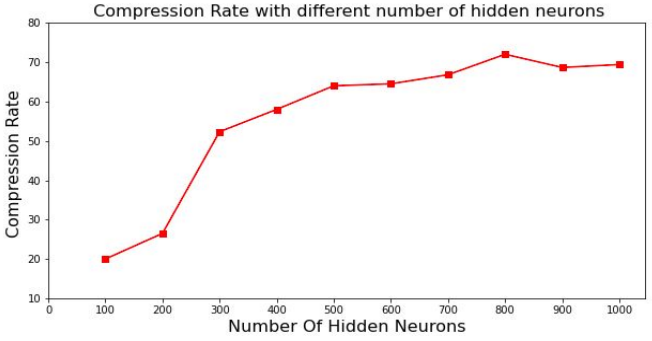

Fig.5 Compression rate with different number of hidden neurons. Compression rate increased as the number of hidden neurons increased.

3.4 Effect of pruning on the model

According to our observations and results obtained so far, the test accuracy obtained with 200 hidden neurons is the best. So we would expect to see a decrease in performance measure when we prune the networks that have less than the optimal number of hidden neurons initially. On the other hand, we would expect an increase in the performance measure when networks having large numbers of hidden neurons are pruned. Whenever networks with large numbers of hidden neurons are reduced to approximately 200-300 neurons, the performance measure should likely be better as the model has removed most of the undesired parameters and attains better generalization. From Fig.6 it is evident that our hypothesis was indeed precise and the results from our experiments made our assumptions valid. Overall the pruning technique to an extent was useful in removing undesired and redundant neurons and increasing the performance of our model with a large number of hidden neurons.

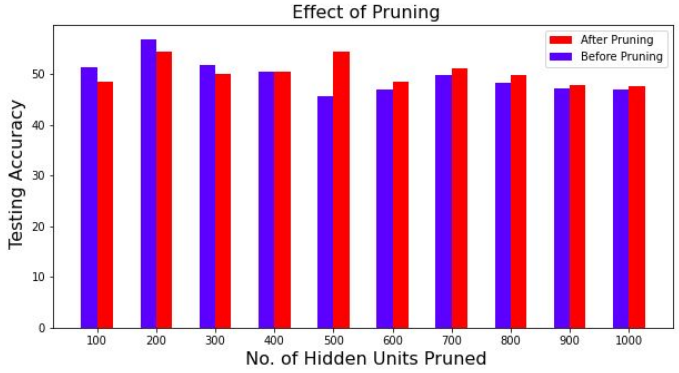

Fig.6 Effect of pruning on different numbers of hidden neurons.As the number of hidden neurons increased , the model after pruning performed better than the original model.

3.5 Classification Report And Analysis

The Classification Report that provides average precision and recall for each class (emotion labels) was the most important evaluation measure in the SFEW dataset and was used as a benchmark.From the results obtained in our previous section we have observed that as the number of hidden neurons increased the model after pruning performs better than the original model. But, for our optimal model with 200 hidden neurons, pruning has a negative effect as it decreases the performance of our model as a lot of necessary hidden units might have been removed during the pruning process which led to a decrease in the performance measure. Hence,we only show the results of our optimal model and how pruning affects this model.

BEFORE PRUNING							
Emotion	**Angry**	**Disgust**	**Fear**	**Happy**	**Neutral**	**Sad**	**Surprise**
Precision	0.44	0.67	0.44	0.57	0.50	0.73	0.57
Recall	0.40	0.40	0.57	0.73	0.44	0.53	0.73
f1-Score	0.42	0.50	0.50	0.64	0.47	0.62	0.64

Table 4. Average expression class wise Precision, Recall and f1-score results before model pruning.

Table 4 shows the classification report before pruning our optimal model(200 hidden neurons) .Note that models with large numbers of hidden neurons would achieve higher performance measure after pruning which is not the case for small numbers of hidden neurons as mentioned earlier in the paper.

AFTER PRUNING							
Emotion	**Angry**	**Disgust**	**Fear**	**Happy**	**Neutral**	**Sad**	**Surprise**
Precision	0.56	0.67	0.33	0.57	0.38	0.91	0.36
Recall	0.62	0.22	0.38	0.67	0.33	0.62	0.83
f1-Score	0.59	0.33	0.35	0.62	0.35	0.74	0.50

Table 5. Average expression class wise Precision, Recall and f1-score results after model pruning.

From Table 4 and Table 5 we can observe that the model after pruning achieves less overall precision than the model before pruning. This was discussed in section 3.4 where our hypothesis turned out to be accurate.

4 Conclusion and Future work

From our research and observations they were many conclusions that we ended up with:

- From the results obtained for the simple feed-forward neural network it was observable that when the number of neurons pruned are more than 8 (out of 20 that we started initially with) our reduced network performs better than the original model. But when less neurons are pruned (between 0 or 8) the original model performs better on the test set.It concludes the fact that 20 hidden neurons is eventually larger than the optimal number of hidden neurons since the network reduces by 40% almost every time.
- Having 200 hidden neurons seemed to be optimal for our model as the test accuracy peaked at 200 hidden neurons with 56.84%.It was observable that as the model complexity got higher and higher (large number of hidden neurons) the model tended to overfit more and was not able to generalize better.
- The reduction rate of our network increased as the number of hidden neurons increased. For 200 hidden neurons the reduction rate was 26.5% whereas for 1000 hidden neurons the reduction rate was 69.4%. From our results it was noticeable that the optimal number of hidden neurons our models reduced to was approximately between 200-300 neurons.
- Overall the pruning technique to an extent was useful in removing undesired and redundant neurons and increasing the performance of our model with a large number of hidden neurons.But on the other hand when the number of hidden neurons are small the pruning had negative effect on the performance measure as a lot of necessary hidden units might have been removed during the pruning process which led to a decrease in the performance measure.

The images that the SFEW dataset contains are not the typical images that a CNN processes.Even after running face detection the cropped images of the faces are not aligned and centred as per the basic standards used.The images are abrupt in the sense that emotion in some faces are not clearly visible as the faces are not constant and have varied head poses and different focus.If all the faces are perfectly aligned and centred and made similar to 'ImageNet' standards then attempting transfer learning with better pre-trained models such as Vgg-16 , GoogleLeNet etc would have increased the performance measure.We only experimented with the standard threshold proposed by Gedeon and Harris[1] but if we varied the threshold angle of distinctiveness according to the number of hidden neurons we might have ended up with different results.Setting a more larger threshold for small number of hidden neurons and lowering the threshold for larger number of hidden neurons can be experimented with in the future.In addition to this, we have only fine tuned the fully connected layers of the ALexNet according to our dataset, if we try freezing only some layers of convolution and then train our model on the fully connected layer plus some final convolutional layers of our own than the predictive performance of the model should increase.

5 References

[1] Gedeon, TD, Harris, D, “Network Reduction Techniques,” Proc. Int. Conf. on Neural Network Methodologies and Applications, AMSE, vol. 1, pp. 119-126, San Diego, 1991a. Proc. 9th Ann. Cog. Sci Soc. Conf. , Seattle, pp. X-y, 1987.

[2] Dhall, A., Goecke, R., Lucey, S., & Gedeon, T. (2011, November). Static facial expressions in tough conditions: Data, evaluation protocol and benchmark. In 1st IEEE International Workshop on Benchmarking Facial Image Analysis Technologies BeFIT, ICCV2011.

[3] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Acted Facial Expressions in the Wild Database. In Technical Report,2011.

[4] Sheela, K.G. & Deepa, S.N.,: Review on Methods to Fix Number of Hidden Neurons in Neural Networks,Mathematical Problems in Engineering, vol. 2013, pp. 1-11. 2013

[5] Z. Zhang, "Improved Adam Optimizer for Deep Neural Networks," *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, Banff, AB, Canada, 2018, pp. 1-2.

[6] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[7] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[8] Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In *NIPS*.