Convoluted Cuts:
Distinctiveness Pruning on Convolutional Kernels

George Breynard[1]

Research School of Computer Science[1]
Australian National University, Australia

Abstract. The pruning of neural networks has become increasingly important as a method of deploying usually resource-intensive convolutional neural networks in resource-constrained real-world environments. This paper investigates the effectiveness of *distinctiveness* (as a measure of similarity between two kernels) in pruning techniques for convolutional neural networks. It does this by comparing the effectiveness of an extended *distinctiveness* pruning algorithm at maintaining network functionality to that of a random pruning algorithm over ten trials. The extended *distinctiveness* pruning algorithm was consistently found to outperform random pruning, on average maintaining 10% more accuracy and performing even in some trials. This provides direction for further research into the use of distinctiveness as a measure of similarity for pruning.

Keywords: Machine Learning, Neural Networks, Pruning, Pruning Algorithms, Distinctiveness, CNN, Emotion Detection

1 Introduction

The pruning, or systemic removal, of neurons from neural networks has become an increasingly popular focus for machine learning research in recent years. Research activity has to an extent been driven by the necessity of deploying evermore resource-intensive deep neural networks in resource-constrained real-world environments. Judicious and prudent pruning has the capacity to produce smaller, less resource intensive neural networks whilst preserving a similar level of accuracy (Blalock et al., 2020). However, creating a metric that accurately assesses the importance of a specific neuron to a networks function is quite a challenge. Pruning is particularly relevant to the deployment of Convolutional Neural Networks as they are on average more resource intensive to train and run than other forms of neural network (Lu et al., 2017).

This paper seeks to extend the pruning technique of using the *distinctiveness* of a network's neurons to determine which should be pruned, introduced in Gedeon & Harris (1991) and described in Gedeon & Harris (1992), to prune the kernels of a Convolutional Neural Network (CNN). This technique will be evaluated by comparing how a baseline network's functionality decays as it is pruned using this extended *distinctiveness* pruning technique, versus how its functionality decays as kernels are pruned randomly.

1.1 Distinctiveness Pruning

Most automated pruning algorithms use a measure of each neuron's importance to the overall network's function to assess whether any particular neuron should be pruned from the network. The *distinctiveness* of each neuron is one measure that is used to compare the function of two neurons (Gedeon & Harris, 1992). To calculate *distinctiveness* first an activation vector is calculated for each neuron with dimensionality equal to the number of elements in the training set. Each component of this activation vector corresponds to the activation of that neuron after each sample from the training set is passed through the network. The *distinctiveness* between neurons is then calculated as the angle between their activation vectors. So, for activation vector u of a and activation vector v of b:

$$distinctivness(a,b) = \frac{u \cdot v}{|u|_{\Gamma}|v|}$$

Gedeon and Harris' original paper on this technique proposes three classes of undesirable neurons that can be pruned from the network based on their distinctiveness (Gedeon & Harris, 1992). Pairs of neurons with an angular separation of less than 15°, which perform a very similar function can be pruned by removing one and adding its weight vector to that of the remaining neuron. Pairs of neurons with an angular separation of more than 165°, which are complementary can both be removed without further adjustment. Where the sum of the activation vectors of a group of neurons is zero or constant, the group has no or constant effect and can be removed where the sum is zero, or replaced by a constant bias term otherwise, after which it may be necessary to retrain the network.

1.2 Convolutional Neural Networks

CNNs are a type of deep neural network that utilize convolutional kernels to extract features from tensor inputs, commonly 3-channel RGB images (Sakib et al., 2018). Convolutional kernels are usually applied as a convolutional layer to a network's input. A convolutional layer is composed of multiple convolutional kernels, tensors of smaller dimension than the input tensor, that are applied to different regions of the input tensor by calculating the corresponding dot product. As a kernel is applied to different regions of the input tensor, the distance between which is determined by the kernel's step size, it generates an activation map of how similar the kernel tensor is to each of these regions.

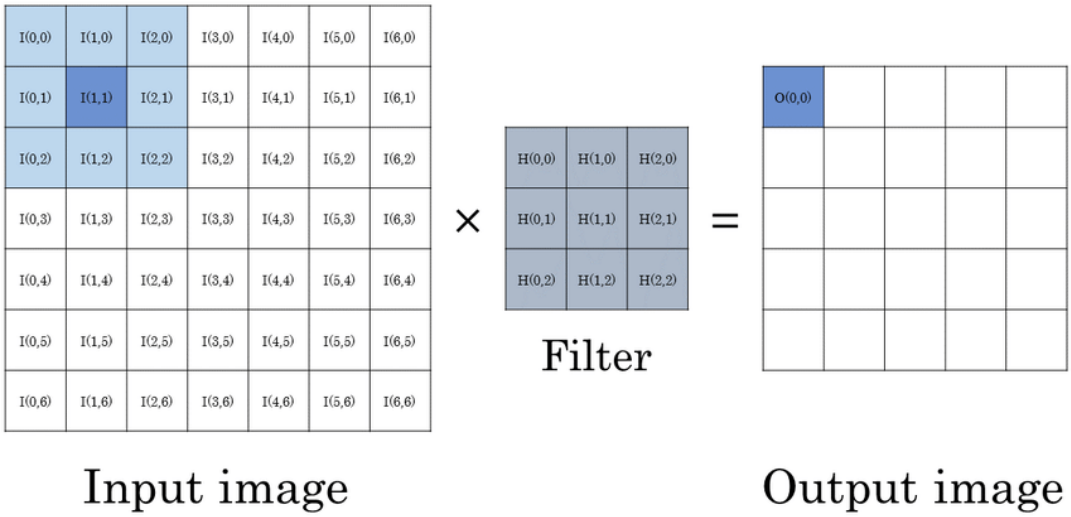

Fig. 1. Convolutional Kernel Diagram (Baskin et al., 2017)

As the CNN learns these kernels, they allow it to detect certain features within the input tensor, such as edges in an image tensor. CNNs commonly utilize multiple convolutional layers, pooling layers which reduce the dimension of an input tensor, convolutional padding which helps maintain the dimension of an input tensor by padding it with zeros and linear layers to combine these features, to produce an output. Convolutional kernels allow CNNs to better interpret complex data and the structures that exist within it than basic neural networks that can only handle linear input.

1.3 Distinctiveness Pruning on Convolutional Neural Networks

Since originally *distinctiveness* pruning was designed to be applied to a basic neural network the concept of *distinctiveness* needs to be extended to be applied to the convolutional kernels in a CNN. The activation vector of a kernel will be an N dimensional vector, where N is the number of samples in the training set, and each entry of the activation vector corresponds with the Frobenius norm of the activation map of that kernel after each sample is passed through the network. Then the *distinctiveness* between kernels can be calculated as the angle between their activation vectors, similar to the original algorithm, yielding the equation:

$$distinctiveness(A, B) = \frac{\langle U,V \rangle}{|U|\,|V|}$$

Where U is the activation vector of kernel A and V is the activation vector of kernel B. For the purposes of this paper simpler conditions than those proposed by Gedeon and Harris will be used to determine which kernels to prune based on their *distinctiveness*. This extended *distinctiveness* pruning algorithm for convolutional kernels will, each time it is called, prune one kernel from the least distinct pair of kernels. This means the algorithm will be deterministic in the sense that from a starting network the same kernels will always be removed in order of increasing *distinctiveness*, measuring the effectiveness of using this new equation for kernel *distinctiveness* as a measure of a kernel's function, rather than extending the entire *distinctiveness* pruning algorithm to a class of neural network it was not designed for.

2 Method

In order to test the effectiveness of our new extended definition of *distinctiveness* as a measure of a kernel's function a baseline CNN model was first created and trained on the Static Facial Expressions in the Wild (SFEW) dataset (Abhinav Dhall et al., 2011). Then, using the extended *distinctiveness* pruning algorithm for convolutional kernels detailed above, nine kernels where progressively removed from this baseline network. The resulting network's accuracy on the training and test sets, consisting of 501 and 175 images respectively, were then recorded over ten trials. As a point of comparison, a random pruning algorithm was used on a duplicate baseline network, and as nine kernels were progressively removed the resulting network's accuracy on the training and test sets was recorded over ten trials.

2.1 SFEW Dataset

The SFEW dataset is a collection of static frames from the Acted Facial Expression in the Wild (AFEW) dataset labelled with one of seven expression categories (Angry, Disgust, Fear, Happy, Neutral, Sad, Surprise). These datasets attempt to simulate real world facial expression data by labelling scenes extracted from movies. They are particularly relevant to the task of testing CNNs as they replicate the data these networks normally work with, real-world image data. SFEW images are in the PNG image format so were easily loaded into three channel Python PIL images which could then be converted to [3,576,720] Pytorch tensors. This process also normalized the RGB values of these images. This three-channel normalized image tensor was chosen to train the network on as it maintains the images data structure for CNN to analyze unlike other preprocessing techniques such as flattening the image into a vector. The baseline neural network was trained on a 675-sample subset of the SFEW dataset. Sample images from this dataset can be seen in **Fig. 2.** below.

Fig. 2. Sample images from the SFEW dataset

2.2 Baseline Neural Network

A simple CNN was implemented as a baseline to test the *distinctiveness* pruning algorithm for convolutional kernels on. The baseline network's architecture was as follows: a convolutional layer of nine [3,9,9] kernels with zero padding and a step size of one, a max pooling layer of size nine, a convolutional layer of nine [9,9,9] kernels with zero padding and a step size of one, a max pooling layer of size nine, a flattening layer, a linear layer of 126 neurons with the Rectified Linear Unit (ReLU) activation function, a linear layer of 42 neurons with the ReLU activation function and an output layer of dimension seven; see **Fig. 3.** This architecture was chosen because its simplicity provides a standard baseline on which to test the extended *distinctiveness* pruning algorithm.

.

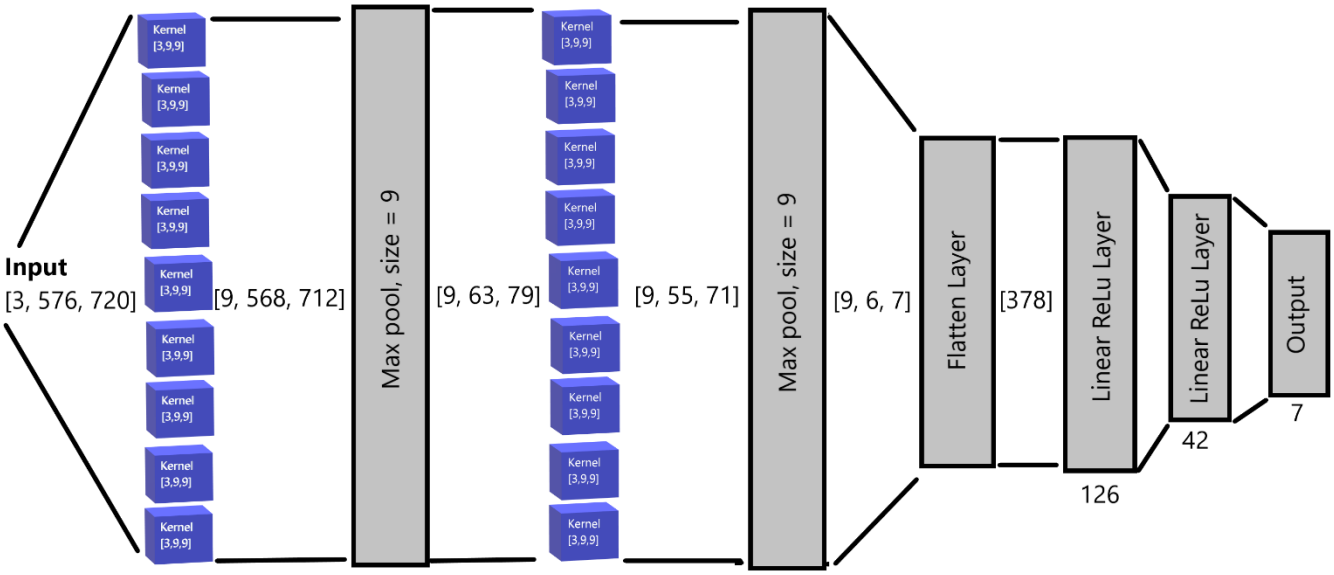

Fig. 3. Baseline Network Architecture

This network was then trained via backpropagation using cross entropy loss with a weight decay of 1e-4 and the Adam optimizer with a learning rate of 1e-2 (Kingma & Ba, 2014). This training took place using minibatches of size 32 over 20 epochs. The network consistently achieved a baseline accuracy of 70-90% on the training data and 35% on the test data with some variation due to the random nature of the network's initialization.

2 Results

The performance of the baseline model on both the training and testing sets was measured progressively as nine kernels were removed from the network in parallel trials: the first using extended *distinctiveness* and the second, random pruning techniques. This was repeated over ten trials and the mean and standard deviation of the data calculated. These results are tabulated in **Table 1.** and **Table 2.** below and are represented graphically in **Fig. 4.** and **Fig 5.**.

Table 1. Average and Standard Deviation of the baseline CNN's accuracy on the test and training sets after nine neurons are progressively removed via the random pruning algorithm over ten trials.

Kernels Pruned (#)	0	1	2	3	4	5	6	7	8	9
Accuracy on test set	28	28	26	26	24	20	20	17	21	20
Deviation on test set	5	5	4	3	4	2	4	4	3	1
Accuracy on training set	88	86	72	71	67	54	46	38	36	29
Deviation on training set	10	10	24	23	18	14	13	18	13	15

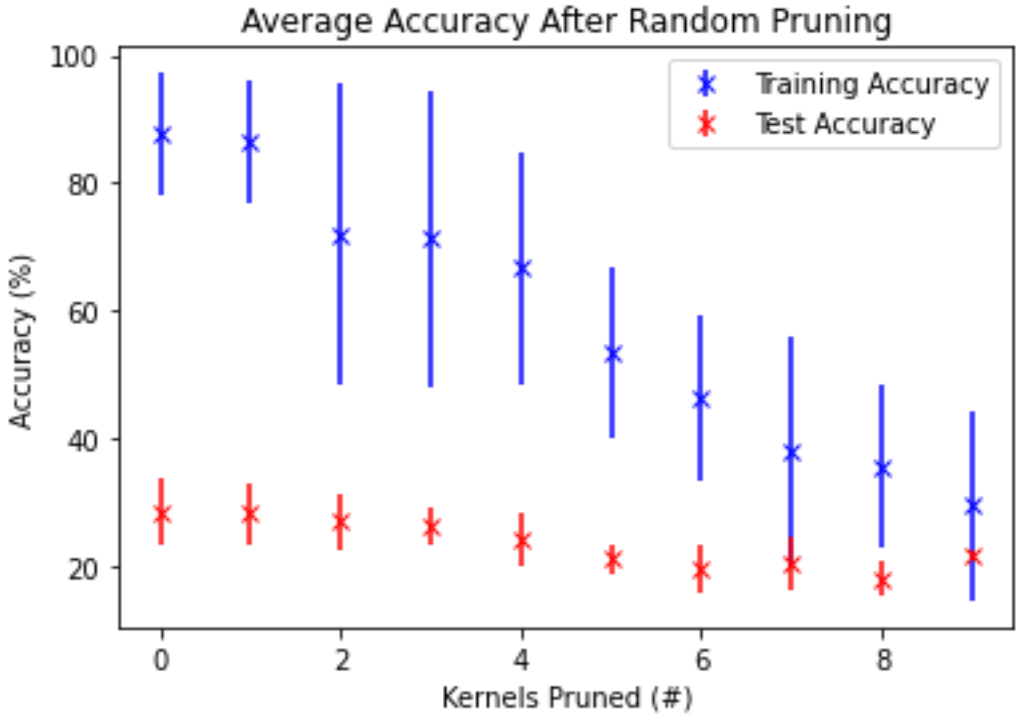

Fig. 4. Graphical Representation of results in **Table 1.**

Table 2. Average and Standard Deviation of the baseline CNN's accuracy on the test and training sets after nine neurons are progressively removed via the *distinctiveness* pruning algorithm over ten trials.

Kernels Pruned (#)	0	1	2	3	4	5	6	7	8	9
Accuracy on test set	37	34	34	29	22	23	22	21	19	20
Deviation on test set	4	5	7	9	5	4	5	7	10	8
Accuracy on training set	91	81	84	73	77	56	52	54	41	41
Deviation on training set	6	15	11	26	13	25	24	24	29	29

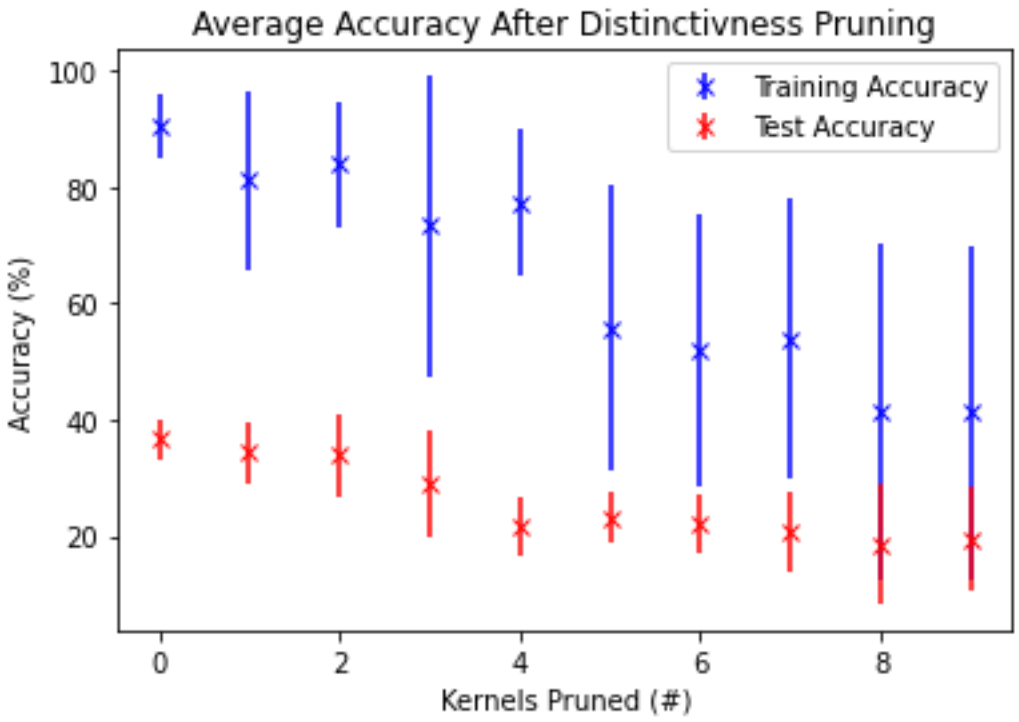

Fig. 5. Graphical Representation of results in **Table 2.**

4 Discussion

The results above provide insight into how each pruning technique, random pruning and extended *distinctiveness* pruning, effect the baseline network's accuracy on both the training and test datasets. An immediate observation to be made is that although some features remain consistent, considerable variation was recorded in the effects of each technique on the baseline network's performance. This may point to the significant impact a network's initial structure has in determining the effect the application of a pruning algorithm has on its accuracy. Another general observation that can be made is that in general the standard deviation of results on the test data set are always lower than that on the training data set.

4.1 Comparison of Pruning Techniques

As a point of comparison, the effect of random pruning on the accuracy of the baseline network will be discussed first. It appears that initially randomly pruning one or two kernels only results in a moderate variation in accuracy, at worst dropping just below 80% for the training set. Then after the second kernel is pruned results seem to diverge greatly as the standard deviation of the data increases. Finally, after five neurons are randomly pruned the deviation in trials grows much smaller and network accuracy more consistently trends downwards.

The behavior of the extended *distinctiveness* pruning algorithm on network accuracy although initially appearing quite similar to that of the random pruning algorithm differs in some important ways. The *distinctiveness* pruning algorithm seems consistently to be able to prune up to two kernels without a significant effect on network accuracy, keeping the average training accuracy above 80%. After this point accuracy slowly declines with a much greater variation in results. Some trials maintained a high level of accuracy after nine kernels were pruned and others rapidly declined in accuracy after the fourth kernel was pruned.

Comparing these two pruning techniques highlights interesting behaviors in each. It appears that random pruning generally results in a much more rapid and consistent decline in network accuracy. *Distinctiveness* pruning on the other hand appears to maintain a higher level of accuracy for longer and although for some networks as more kernels are pruned the results are similar to random pruning, for others a much higher level of accuracy is maintained than was the case with random pruning. Averaging the gap between results quantifies this difference, demonstrating that *distinctiveness* pruning retains on average 10% more network functionality than random pruning.

4.1 Findings

These observations appear to support the effectiveness of using *distinctiveness* as a measure of similarity between kernels as a means of optimizing network accuracy via selective pruning. Three clear trends can be drawn from these observations: the baseline network likely started with a greater number of convolutional kernels than needed, *distinctiveness* pruning maintains greater network functionality/accuracy as kernels are pruned and the effectiveness of *distinctiveness* pruning is highly dependent on the initial network. Two observations support that the network started with a greater number of kernels than needed: firstly, even random pruning could consistently remove one or two neurons without a large decrease in network functionality meaning that a number of effectively redundant kernels were present, and secondly, the large gap between training and test set accuracy is a likely indicator of overfitting i.e. the network being overly complex. Over the ten trials conducted the *distinctiveness* pruning algorithm resulted in a much slower average decrease in network accuracy than the random pruning algorithm as well as a much higher deviation as kernels were pruned, indicating that some networks maintained a high level of functionality even after nine kernels were pruned. Finally, the high deviation in the results of the *distinctiveness* pruning algorithm, especially as greater numbers of kernels were pruned, points to network architecture having a high correlation with the degree of functionality maintained after pruning. This result makes intuitive sense as some networks may be dependent on a greater/smaller number of kernels, or work may be unevenly distributed across the network structure, and therefore different network architectures will respond differently to kernel pruning.

5 Conclusion and Future Work

Using a simple CNN as a baseline it has been shown that the extended *distinctiveness* pruning algorithm for kernels consistently outperforms a random approach to pruning, maintaining an average of 10 % more accuracy in networks it is applied to. This result is apparent from several trials which show that the extended distinctiveness pruning algorithm results in a much slower decay in network functionality as kernels are removed and that some networks can tolerate the removal of many kernels and still maintain a good level of functionality. Conversely, pruning a kernel using the random pruning algorithm consistently led to a much more rapid and uniform decrease in network functionality across all trials. These results support the effectiveness of distinctiveness as a measure of the similarity between neurons which may have many potential useful real-world applications, including in the pruning of deep neural networks for use in resource-constrained environments. It must still be noted however, that experimentation was only conducted on one CNN architecture and using a relatively small sample size of ten trials. Possible areas of future research in this field are a more exhaustive assessment of the extended *distinctiveness* pruning algorithm detailed in this paper to assess its effectiveness on a range of network architectures, and the proposal of alternate pruning algorithms which utilize distinctiveness as a measure of a kernel's function.

References

Baskin, C, Liss, N, Mendelson, A, Zheltonozhskii, E, "Streaming Architecture for Large-Scale Quantized Neural Networks on an FPGA-Based Dataflow Platform", Arvix, 2017, Accessed via https://arxiv.org/abs/1708.00052 on 31.5.21

Blalock, D, Ortiz, JJG, Frankle, J, Guttag, J, "What is the State of Neural Network Pruning?", Arvix, 2020, Accessed via https://arxiv.org/pdf/2003.03033.pdf on 27.4.21

Dhall, A, Goecke, R, Lucey, S, Gedeon, T, "Static Facial Expression in the Wild", Australian National University, 2013 Accessed via https://cs.anu.edu.au/few/AFEW.html on 27.4.21

Gedeon, TD, Harris, D, "Network Reduction Techniques", Int. Conf. on Neural Networks Methodologies and Applications, AMSE, San Diego, 1991.

Gedeon, TD, Harris, D, "Progressive Image Compression" IJCNN International Joint Conference on Neural Networks, Baltimore, MD, USA, 1992.

Kingma, PD, Ba, J, "Adam: A Method for Stochastic Optimization", Arvix, 2017, Accessed via https://arxiv.org/pdf/1412.6980.pdf on 27.4.2

Lu, Z, Swati, R, Chan, K, Porta, TL, "Modeling the Resource Requirements of Convolutional Neural Networks on Mobile Devices", Arvix, 2017, Accessed via https://arxiv.org/pdf/1709.09503.pdf on 31.5.21

Sakib, S, Ahmed, N, Kabir, A, Ahmed, H, "An Overview of Convolutional Neural Network: Its Architecture and Applications", Dept. of EEE, Independent University of Bangladesh, 2018.