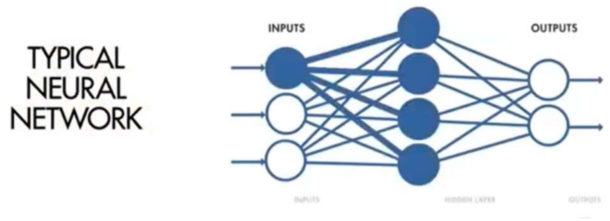
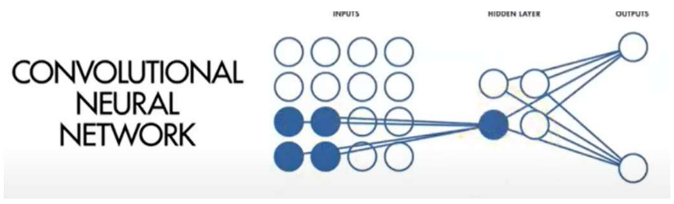
Analysis of CNN Model Architecture and Parameters

In this lab, I worked on the same project notebook file, “Chichua or Muffin”, from assignment L06. This will take it a step further by changing the parameters and architecture to see if this Convolutional Neural Network can give me better results with less error and higher accuracy.

In the previous workshop, “TensorFlow Playground Presentation,” I worked with traditional neural networks as we can see in the picture below. Each neuron in the input layer is connected with the neurons in the hidden layer and it applies to the all the layers in the traditional neural networks. It is good for simpler tasks where the layout of the data doesn't really matter.



In Convolutional Neural Networks, as you can see in the figure below, only the small region of the input layer neuron connects to the neuron in the hidden layer. That small portion is called the local receptive field. Now this field translates across the image so we can create a feature map from the input layer to the hidden layer neuron and this is done with the help of convolution method. Also, in CNN, a shared weight and bias is used. This means that the amount of weights that have to be trained is smaller compared to the weights that are required to be trained in a traditional neural network. CNNs also automatically pick out features like edges and textures in the input images or videos. They are very efficient for image and spatial data tasks because they share parameters and focus on local areas, making them faster and more effective for these types of problems.



In the previous assignment, I added the height and width of the image, batch size and some other information that was missing from the code. With the given parameters used, the results were not that accurate. But in this assignment, I was able to change the parameters and layers of the CNN which gave me much better results. I changed the following parameters from the CNN model. I added and deleted layers, changed the learning rates, and changed the epoch count. I saw that accuracy was higher (more than 90%) and errors were less (less than 40%) than with the base parameters from the previous assignment.

Below is a table of the results from changing various parameters in the model. The first test that I did was to increase the epoch count (rows 2 and 3 below). From the data below, it shows that increasing the epoch count increases the training accuracy and reduces the training error. Initially, I expected the same result from the validation accuracy and error. This was not the case, as accuracy decreased, and error increased going from 50 to 100 epoch count. The next test I performed was increasing the learning rate (rows 4, 5, and 6). Increasing the learning rate from 0.1

to 0.5 does not make a big difference to the accuracy and error levels. Increasing the learning rate from 0.5 to 0.8 provides better results with better accuracy and error for the training and validation data. Increasing the learning rate to 1 gives mixed results. Training accuracy improves, but training error also increases. A learning rate that is too high may not be beneficial in this case as it overshoots the optimal solution. The next test that I performed was adding a hidden layer (rows 7 to 10). The first case is using the base case parameters, but additional parameters are also changed in the subsequent rows. From the data, there doesn't to be too much difference in the accuracy and error levels until the learning rate is improved. This can be seen in rows 9 and 10 below. The last test that I performed was reducing the number of hidden layers (rows 11 to 14). I found one anomaly in that decreasing by one hidden layer and increasing the epoch was sufficient to drastically better the performance. Subsequent improvements to the learning rate actually makes the results worse. This may be because increasing the epoch when the model is not as complicated (less hidden layers) allows the model to converge better.

	Height	width	Epoch	Learning rate	Hidden layers	Last Train accuracy	Last Train error	Last Validation accuracy	Last Validation error
1.	125	125	3	0.1	same	0.5417	0.6931	0.5667	0.6931
2.	125	125	50	0.1	same	0.9833	0.3308	0.9333	0.4218
3.	125	125	100	0.1	same	1.0000	0.3135	0.8333	0.4583
4.	125	125	3	0.5	same	0.5417	0.5976	0.5667	0.6684
5.	125	125	3	0.8	same	0.8000	0.5336	0.8000	0.5144
6.	125	125	3	1	same	0.8333	0.5755	0.7000	0.6173
7.	125	125	3	0.1	+1	0.5417	0.6931	0.5667	0.6931
8.	125	125	10	0.1	+1	0.5417	0.6931	0.5667	0.6931
9.	125	125	10	0.5	+1	0.9583	0.3570	0.9000	0.4132
10.	125	125	10	0.8	+1	0.9250	0.3879	0.9333	0.3865
11.	125	125	3	0.1	-1	0.6667	0.6165	0.7667	0.5455
12.	125	125	10	0.1	-1	0.9750	0.3501	0.8333	0.4542
13.	125	125	10	0.5	-1	0.6167	0.6851	0.6000	0.6821
14.	125	125	10	0.8	-1	0.5417	0.5277	0.5667	0.5552

Compared to traditional neural networks, CNNs will have a shorter training time for the same data. Traditional neural networks require a large number of parameters because the network is fully connected. CNNs, on the other hand, are only connected in the local receptive field. In terms of performance, CNNs excel at image and video recognition tasks because they can efficiently capture spatial and temporal dependencies due to their local receptive fields and shared weights. On the other hand, traditional neural networks treat every input feature as independent and do not take into consideration the spatial structure of the data. This makes them less effective for images and videos.

One of the challenges that I faced during this assignment is understanding and analyzing the results. There were many times where the results were mixed and not trending in the expected direction. There were some learnings out of this, especially that a high learning rate may not always be beneficial. Additionally, increasing the epoch count seems to be generally beneficial to improving the accuracy and error levels. The other challenge that I faced was understanding why the number of features in the output of a hidden layer is cut in half compared to the input. After some research online, it seems that this is beneficial to help simplify the model and improve computational efficiency. I followed the same approach when I was adding and subtracting hidden layers from the model.

There are many real-world applications of this type of convolutional neural network. One of them is object detection for self-driving cars such as detecting pedestrians and traffic signs. Another is facial recognition for security systems, social media tagging, and payment verification. Another category is medical imaging, such as disease diagnosis and image segmentation. Image segmentation is where certain regions/areas of a medical image are isolated to detect anomalies better. Another category is retail and e-commerce. CNNs can be used to determine similarities in products to help better categorize and recommend as well as use images to help search for available products that are similar. There are a lot of applications possible with CNNs and only a few are highlighted above.

One of the ethical concerns with CNN models is the potential bias in the data that is used for training. One example is facial recognition systems that have been mainly trained on lighter-skinned people. Another concern is data privacy and security as these models are highly dependent on data, which may not have been consented prior to use. Lastly, as for every neural network, they all are like a black box and understanding their conclusions are not always straightforward. Too much reliance on this can prove to create some backlash, especially if the conclusions of the model conflict with people's belief and preferences.

Reference:

Hussain, Mahbub & Bird, Jordan & Faria, Diego. (2018). A Study on CNN Transfer Learning for Image Classification.

M. Shaha and M. Pawar, "Transfer Learning for Image Classification," 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2018, pp. 656-660, doi: 10.1109/ICECA.2018.8474802.

Course materials –
ITAI 1378 2024 Mod 04 Image Fundamentals of Image Processing.pptx
ITAI 1378 2024 Mod 05 ML for Computer vision.pptx
ITAI 1378 2024 Mod 07 Convolutional Neural Networks.pptx