Presentation Transcript

<u>Slide 1)</u>

Welcome to my presentation on the creation of a neural network using the cifar-10 image dataset for object recognition. This presentation will explain the various steps I took to create, design, and evaluate a neural network.

<u>Slide 2)</u>

When working on an assignment, the first step that must be undertaken is ensuring that you have fully understood what it is you're being asked to do. In this assignment, the student is asked to develop a neural network using the CIFAR-10 image dataset for object recognition. The primary objective of the assignment is to design the neural network and assess the model's efficacy. An oral presentation was to be created in order to fully convey understanding of the assignment, as well as demonstrating the ability to convey to others the knowledge gained during the undertaking of the assignment.

The learning outcome of this assignment was to be able to "apply and critically appraise machine learning techniques to real-world problems, particularly where technical risk and uncertainty is involved". I hope to demonstrate in this presentation that this objective has been achieved.

<u>Slide 3)</u>

Now, the data to be used to develop the neural network has already been mentioned to be the CIFAR-10 dataset. But what exactly is the CIFAR-10 dataset?

The CIFAR-10 dataset is a subset of the “80 million tiny images” dataset. The data was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton (Krizhevsky, 2009). The dataset contains 60,000 images, with each image being 32 pixels wide and tall. The dataset contains 10 classes which describe what the object is. The classes consist of the following: airplane, automobile, cat, deer, dog, bird, frog, horse, ship, and truck. Each class contains 6000 images, and they are mutually exclusive. No airplane image can belong to any other class, and a truck is not the same as an automobile.

Slide 4)

Here you can see an example of the various kinds of images contained in the dataset. The images are very small, and do not have a lot of detail. Regardless, we hope to be able to train our neural network to be able to tell the difference between a bird and a plane!

Slide 5)

Next, we need to think about how to partition the data. Luckily, the CIFAR-10 dataset is already split into training batches and test batches, with 50,000 images being in the training batch and 10,000 in the test batch (Krizhevsky, 2009). The data was first divided into the test batch, with exactly 1,000 randomly selected images from each class being put into the test batch. The training batches are made of all the remaining images in random order, and the training batches most likely contain varying amounts of each class. While the division into training and test batches was done for us, the next few steps are our own work.

Slide 6)

First of all, we needed to ensure that the data was correctly preprocessed to ensure consistency and improve the performance of the model. First, we normalized the data. While not strictly necessary, researchers such as Sola and Sevilla (1997) have stressed that it is very important to normalize data prior to the training process in order to obtain quality results and reduce the calculation time. The CIFAR-10 dataset comes in the form of images. This means the data is stored in pixels, and pixel values come in a range of 0 to 255. To normalize the data, we effectively divide the training and testing set pixel values by 255 to normalize the values between a range of 0 and 1.

After this, some further preprocessing was necessary. The various classes the CIFAR-10 dataset images come in are categorical and would be best stored as a binary matrix representation. While this can be done in a few different ways, such as binary encoding, one-hot encoding appears to perform best in machine learning models (Cedric, 2018). After preprocessing the data, one last step had to be done as a part of the data partitioning process. This was to create a validation set based off of the training data. The validation set is a randomly picked training batch that will be used later on in the training process of the model.

<u>Slide 7)</u>

But why is it important that we have a validation set? Couldn't we just train the data using the 5 training batches, instead of reserving one as a validation set? What purpose does a validation set have? These were all questions that I had when I first approached this assignment. The main reason to create a validation set is to prevent overfitting. The entire purpose of creating a neural network for object recognition is to be able to feed the neural network new images and have it be able to predict what that image is on its own after practicing on the training data. Hawkins (2003) describes overfitting as the violation of parsimony, or including more data than is strictly necessary. When this happens, the model may struggle to accurately tell the difference between a bird and an airplane when given a new image that is not a part of the training data.

We use a validation set to "provide an unbiased evaluation of a model fit on the training dataset", as stated by Brownlee (2020). You might wonder how this differs from a training set. I definitely did. And in fact, this is not an uncommon phenomenon. The literature in the field regularly confuses the two topics, and even introduces other similar concepts such as a "hold-out set" (James et al, 2013). The difference between a validation set and a test dataset is that a validation set is used while tuning the model hyperparameters and training the model. The test dataset provides an evaluation of the final model and cannot be used before the model is finished training.

<u>Slide 8)</u>

Now that we have our data partitioned and ready to use, we need to come up with a model design. There are many different kinds of neural networks, however I chose to create one using ResNet due to the module seminar. Originally, I had created a simple Convolutional Neural Network, however after doing some research and watching the seminar, I was convinced that ResNet was a superior option. While ResNet is a form of CNN, when referring to CNN from now on I will be referring to a simple model that does not overcome the “vanishing gradient” problem that ResNet overcomes. My original model had a 70% accuracy, and I wanted something a bit better than that.

ResNet appeared to be the solution. Miao and his colleagues (2021) argue that ResNet is superior to CNN, and that ResNet has a “better learning ability and improved performance in image recognition tasks”. As creating a neural network capable of object recognition is the objective of our assignment, ResNet seems to be our best option. While creating the ResNet model, I did notice that it took significantly longer to calculate the model. While not necessarily needed information for this presentation, it was interesting to learn. If you don’t need to be too accurate or don’t have the computing power to create a powerful model, then a simple CNN might be preferred over a ResNet model.

Slide 9

However, I struggled with the creation of the ResNet model. I could not figure out what went wrong with the creation of it, and so decided to go back to my original neural network that I built. While disappointed, this is a part of the learning process and I decided to keep it in my presentation to acknowledge the fact that not everything goes according to plan. I will be uploading both notebooks to the assignment, and I'd be extremely appreciative if you could tell me what I did wrong, as I am completely blind to it. However, it is not necessary if you do not have the time! And now, on with the presentation.

Slide 10

A convolutional neural network has several layers. The first of which is the input layer. This layer is where the image first gets fed into the neural network. As the size of the image is 32 pixels by 32 pixels, and we are using a RGB scale, we ensure the input channel has 3 channels for each color. The convolutional layers are what makes a convolutional neural network what it is. These layers are what distinguish the images from each other. They extract features from the images and create a feature map to slowly build an understanding of what makes this image different from the others. How do the features of a bird differ from that of a plane? That is what the convolutional layer distinguishes. The pooling layer reduces the spatial dimension of the feature maps created in the convolutional layers. This model uses max pooling, which selects the maximum value within each feature map as the output of that layer (Zhao & Zhang, 2024). The last few layers flatten the input tensor to a vector, and then the chosen activation functions are activated before the final output.

The chosen activation functions were the rectified linear activation function or "ReLu" and the Softmax activation function. ReLu adds non-linearity to the neural network. ReLu has an advantage over other similar activation functions as negative input values give a result of zero, which makes ReLu much more computationally efficient compared to its major competitors, sigmoid and tanh (Gupta, 2024). While Relu is used to make neural networks learn more complex relationships between the input and output data, Softmax normalizes the input values, which helps in classification problems such as object recognition, and it is used in the last layer of the neural network to predict the class of an input image (Saxena, 2014). These functions appeared to be the most helpful when it comes to creating a neural network that could identify an object's class accurately and efficiently.

Slide 11

The data training process originally had me a bit confused. First, I chose a loss function, which ended up being categorical_crossentropy. This was implemented due to the fact that our model has 10 output labels, which are one-hot encoded. When this loss function is implemented, the output label is converted into categorical encoding (Kuman, 2020). I began training the model after simply choosing on a whim a batch size of 64 and 10 epochs. The model did not end up very accurate, and so I started testing out different batch sizes and epochs just to see what would happen. The model ended up being 70.4% accurate, which barely passes the allowance for what is acceptable. The accuracy of the model was worse to begin with, due to the fact that I used only 10 epochs to train the model. After using different numbers of epochs to various degrees of success, I was curious as to how many I should actually be using. And after some research I discovered "callbacks" from the keras library, which allowed me to program the model to automatically stop at the epoch that was optimal (geeksforgeeks, 2024). This prevented me from overfitting my model by running it on too many epochs, and from underfitting my model by not running it on enough. 22 epochs was the optimal amount for this model. You can see in the diagrams to the left of this slide the validation accuracy and the training accuracy for each epoch, up to 22 epochs.

The batch size that I settled on was 128, which was chosen so that the model would calculate a bit faster while still retaining model quality. However, I thought that my low accuracy might be due to my batch size, so I attempted to run a batch size of 32 to see if that would increase the accuracy of the model. Unfortunately, it did not, and the accuracy ended up at approximately 69%.

Slide 12

To reiterate, the learning outcome of this assignment was to be able to “apply and critically appraise machine learning techniques to real-world problems, particularly where technical risk and uncertainty is involved”. I believe that I have demonstrated this, as I have applied 2 different machine learning techniques to try and achieve the best results possible from the neural network. While I struggled to apply the ResNet technique and ultimately failed to get a good accuracy with it, I still learned a lot through trial and error. Admitting that I did not understand what went wrong and going back to my original neural network to try and increase the accuracy of it taught me quite a bit, such as EarlyStopping to ensure that the model doesn’t overfit and trains on the optimal number of epochs. My model might not be perfect, but it is a good starting point for the first neural network I have built.

I also believe that I demonstrated a good understanding of technical risk and uncertainty, particularly when it came to understanding the importance of a validation set and how vital it is to prevent overfitting. While in the past I had thought that training and testing sets were enough to accurately evaluate the performance of a machine learning algorithm, in the future I will be using a validation set to properly know whether my model will be able to accurately predict future outcomes outside of the training data.

Overall, this assignment was a wonderful way to apply machine learning techniques to real-world problems and understand the importance of machine learning in data science in a much more applied way than I have in my past modules. I have only begun to scratch the surface of what can be done with data and programming, and I’m eager to apply my skills and learn more.

Slide 13

Here are the references I used throughout the presentation. Thank you very much for your time, and I hope that you found this presentation enjoyable.