**Development Individual Presentation Project Transcript**

**Convolutional Neural Network Model Training using MNIST Dataset**

Hello everyone, my name is Simbiso Makunde, and welcome to my presentation on developing a Convolutional Neural Network Machine Learning Model which can classify the images of handwritten digits using the MNIST dataset.

In this project, I focused on training a machine learning model using Python programming language, while explaining how neural networks work at a deeper level. I shall provide some insight into the MNIST dataset, and speak on the Convolutional neural network’s structure or architecture rather, data preprocessing steps, and model design choices. I shall also speak about the techniques that I used to optimize the model, to avoid overfitting and why validation sets are very important when training a Machine Learning Model .Lastly, I shall present the key points of what I learned throughout this process.

**About the MNIST Dataset**

The MNIST or Modified National Institute of Standards and Technology dataset consists of 60 000 28x28 pixel grayscale images in the training set and 10 000 testing images of handwritten digits ranging from 0 to 9. This also means that there are 10 classes in total which represent these digits. This dataset is very important for data professionals as a benchmark dataset for model training and testing purposes (Activeloop, 2024). My goal is to train a convolutional neural network that can classify these handwritten digits with high accuracy. The uniformity and size of this dataset makes it ideal for such a deep learning task.

**Data Partitioning and Pre-processing**

The first step in creating the machine learning model is to load the dataset into a Python Jupyter notebook and separate a validation set from the existing training sample. I used the train\_test\_split() method from scikit-learn to divide the original 60,000 training images into 80% for training and 20% for validation. This ensured that 12,000 images were randomly set aside as the validation set, while 48% or 48,000 rather remained for model training.

Having a validation set helps us monitor the model's performance on unseen data and to prevent overfitting during model training. According to a 2024 article by Baheti, overfitting occurs when the machine learning model memorizes the pattern in the training data to such an extent that it is unable to classify unseen data. So a validation set is very important when training a model.

Another important step of the Data Pre-processing step or section is to normalize the pixel values by scaling down from 0-255 to 0-1. This helps the CNN train more efficiently. I also reshaped the input data such that the dimension goes from (28 x 28) to (28 x 28 x 1) - which is a 3D format that is more suitable for Convolutional Neural Networks.

Lastly, I used one hot encoding to convert each label into a vector, a format that machine learning models understand. However Shao in 2022 argues that such methods of encoding sample labels may unnecessarily increase the difference between the full probability and zero probability categories, causing the prediction model to rely too much on the predicted category.

**How a Neural Network Works**

Next I shall explain in short how a Neural Network Machine Learning Model functions. Artificial Neural networks are inspired by the human brain, and the process in which they arrive at conclusions is by accepting inputs, identifying phenomena , weighing options and then finally arriving at a conclusion (IBM, 2025)

There are many different types of neural networks, each more suited to certain applications than others. For example Recurrent Neural networks or (RNNs) are more suitable for forecasting and prediction tasks involving time-series data, while Convolutional Neural networks are usually used for pattern recognition and classification tasks.

In a neural network, input data is passed through layers of neurons and each neuron multiplies it by weights, adds bias and then uses an activation function to decide the output to send to the next layer. This continues through the network until it gives a final output, that is, a prediction or a decision. This process is illustrated in the image on this slide to the right, and we can see there that an activation function is very important when determining the output.

**CNN Architecture**

Here I shall discuss the architecture of the Convolutional Neural network that I designed for this image recognition task using the MNIST dataset. I opted to use the CNN because as previously discussed; this type of neural network is widely used for classification and image recognition tasks. Seng et al state in their 2021 paper that recent research works have also implemented convolutional neural networks in facial recognition task, document analysis and speech detection.

The CNN that I designed for this task includes an initial Conv2D layer using 32 (3x3) filters. This layer accepts the input image and detects features like edges and curves in the image. Next is a MaxPooling2D layer which reduces the size of the feature maps to optimize the performance of the model. In the image there you can see an illustration of all the layers of this CNN and how they are organized in relation to each other.

So after the initial layers comes a Dropout layer which randomly turns off 25% of the neurons to improve the generalization capabilities of the network and avoid overfitting. I added a second Conv2D layer, this time with 64 filters to allow the model to be able to detect more abstract and more detailed features within the images. After this layer comes another MaxPooling2D layer and of course another Dropout layer.

The output from these layers is then transformed into a one dimensional array and passed into a Dense layer which has 128 neurons and Rectified Linear Unit or (ReLU) activation function. This fully connected layer assists the network in learning complex combinations of features, and here I also applied another Dropout layer in order to reduce overfitting during training.

The output layer is a Dense layer which has 10 neurons and Softmax activation function. This gives a probability distribution across the 10 classes of digits from 0-9, thereby fulfilling the classification task of image recognition.

**Training the Model**

My convolutional neural network model was trained for 18 epochs using a batch size of 128, and I used categorical cross entropy as its loss function. This function measures the difference between the predicted probability distribution and the true distribution. The loss function measures how well the model is doing, and the smaller the loss, the better the model is performing. During training, the model tries to reduce this loss by adjusting parameters such as weights and biases so that the predictions are more accurate over time.

The importance of maintaining a separate validation set is once again highlighted during the training of the model, because I made sure that after each epoch, I evaluated the performance of the model on the validation set such that when the loss didn’t improve after two epochs or iterations, I could stop training the model. This is a technique known as early stopping, which helped the CNN to avoid overfitting

I trained the model for 18 epochs and at that point the accuracy of the model kind of hit a plateau and we can see in the image at the bottom there that the accuracy at the 18th epoch is 99, 24% while the loss is 2, 98%. I also visualized the accuracy and loss that the model exhibited during training using some charts which I shall present in the next slide.

Last but not least, I used the Adaptive Moment Estimation Optimizer or Adam Optimizer, which adjusts the learning rate for each parameter dynamically.

**Accuracy & Loss Curves**

Now we come to the Accuracy and Loss curves .During the training of the model, both the training and validation accuracy improved steadily, while the loss or loss curves decreased. Here I have two charts showing accuracy and loss versus the number of epochs and we can see there that the training and validation arcs do plateau as the number of epochs increases. The use of dropout, ReLU activation, and early stopping all contributed to good generalization , and we can see here that there was no major gaps between training and validation accuracy curves, indicating that overfitting was effectively controlled.

**Evaluation on Test Set**

When I evaluated the accuracy of the model using the 10000 image testing dataset, the model achieved a Test Accuracy of 99, 29%, demonstrating strong generalization and indicating that the training of the Convolutional Neural Network was successful.

Here the confusion matrix shows where the model struggles to classify the images—for instance, distinguishing between a 5 and a 3 or between a 9 and a 4. These digits can appear quite similar in handwritten form so this is more of an error that arose from the quality of the input images rather than a reflection of the performance of the model. But overall we can see from the matrix that the model correctly clsassifys most of the images.

**Classification Report**

I also generated a classification report which shows a detailed evaluation of the model’s performance across the 10 classification classes. My report has 4 metrics that is precision, recall, f1-score and support.

However, most important is the F1-Score that is a combination of recall and precision which in this case is either 1.00 or 0.99 for each class. We can conclude therefore that the model is performing really well.

I have also included an image showing sample predictions that is the true classes and the predicted classes of 6 images from the MNIST dataset. It is just a visualization of what the image inputs typically look like as well as the true and predicted labels.

**Design Strategy**

The model’s design followed a typical bottom-up strategy. I started with a simple model and increased complexity gradually and each addition was validated using the validation set. Dropout was introduced early, and the ReLU activation function was chosen for its simplicity and speed.

I also researched various methods of visualizing the performance of the model, for example using a confusion matrix, accuracy and loss plots as well as a classification report. And these tools really helped me to monitor the training and performance of my model.

**Reflections and Conclusion**

Through this project, I gained several skills and practical experience in designing and training Convolutional Neural Networks. I saw how different variations of the hyperparameters of the model affected it’s accuracy during training, and the importance of having a validation dataset, and also how techniques like dropout and batch normalization can improve generalization. I also learned the value of visualization in the training of a machine learning model because monitoring loss and accuracy curves helped to keep an eye on the training of the Convolutional Neural Network. The confusion matrix helped pinpoint the cases where the model faced excuse me, some issues with the classification task because of the quality of the images. Most importantly, I learned that convolutional networks are far more effective than basic neural networks when it comes to dealing with image data or when it comes to image classification tasks. So that is quite an important lesson there, in just learning about the architecture of convolutional neural networks, what I can do to improve them as well as how to evaluate them.

Thank you very much for listening; this is the end of my presentation. Please find the list of References in the following Slide.Thank you.

**References**

Activeloop. (2024). Datasets MNIST. Available at: <https://datasets.activeloop.ai/docs/ml/datasets/mnist/#:~:text=What%20is%20the%20MNIST%20dataset,digits%20between%200%20and%209>. [Accessed: 12 April 2025]

Baheti, P. (2024). Train Test Validation Split: How To & Best Practices [2024]. Available at: <https://www.v7labs.com/blog/train-validation-test-set> [Accessed: 11 April 2025]

Bhati, N. (2023). Re: Normalizing a image dataset for CNN? <https://www.researchgate.net/post/Normalizing_a_image_dataset_for_CNN> [Accessed: 11 April 2025]

IBM. (2025). What is a Neural Network? <https://www.ibm.com/think/topics/neural-networks#:~:text=Every%20neural%20network%20consists%20of,network%20is%20Google's%20search%20algorithm>. [Accessed: 12 April 2025]

Seng, L. et al. (2021). MNIST handwritten digit recognition with different CNN Architectures, Journal of Applied Technology and Innovation. 5(1), 2600-7304 DOI: <https://dif7uuh3zqcps.cloudfront.net/wp-content/uploads/sites/11/2021/01/17192613/MNIST-Handwritten-Digit-Recognition-with-Different-CNN-Architectures.pdf>.

Shao, H. et al. (2022). MNIST Handwritten Digit Classification Based on Convolutional Neural Network with Hyperparameter Optimization, Intelligent Automation & Soft Computing. 36(3) DOI:<http://dx.doi.org/10.32604/iasc.2023.036323>