**Technical Implementation Report**

This report provides a technical description of how two neural network models Artificial Neural Network (ANN) and Convolutional Neural Network (CNN) have been integrated to classify images in the Fashion-MNIST dataset. The project was created in the python programming language with the PyTorch in the Google Colab environment.

**1-Data Handling**

**Data Set Loading:** Fashion-MNIST data set that was used in the project was loaded directly through the torchvision.datasets library. The data set that was utilized in the project includes 60,000 training images and 10,000 test images each with 10 classes which were 28x28 grayscale images.

**Data Preprocessing:** Transformation Pipeline: A transformation pipeline was built on the system.Write using transfers. This pipeline was able to generate two basic preprocessing steps:

transforms.ToTensor: PIL images were converted to PyTorch tensors. transforms.Normalize ((0.5,), (0.5,): Scale pixel values in the range [0,1] to the range -1,1]. This normalization is useful in centering the data allowing the network to learn much easier.

**Data Split (Training, Validation and Test):** Since the guidelines required it, the training set of 60,000 images was divided into two sets: training and validation sets. It was divided into 90 percent to 10 percent, and the results were the following:

* Training Set: 54,000 images,
* Validation Set: 6,000 images,
* Test Set: 10,000 images

The torch.utils.data.random split was used to carry out this splitting operation. The validation set is essential in keeping track of the model on unseen data during development, and in isolating overfitting.

In the final stage, a DataLoader with a BATCH\_SIZE value of 64 was used for all three different sets, and the data was fed to the models in batches.

**2-Network Topology (Model Design)**

Both different models were implemented as classes inheriting from nn.Module, and the DEVICE (CUDA or CPU) available at Colab runtime was used.

**Artificial Neural Network (ANN / MLP):** The ANN was developed as a Multi-Layer Perceptron. It has the following overall structure:

* Input Layer: Flattening layer used to turn the 28x28 image into a 784-unit vector.
* Hidden Layer 1: This contains a 256-unit Linear layer directly followed by an activation of ReLU.
* Dropout 1: To add a dropout layer to regularize the model, the p=0.5 dropout was introduced; this is one of the suggested enhancements.
* Hidden Layer 2: 128 unit Linear with an activation of ReLU.
* Dropout 2: A dropout layer has been implemented; p=0.5.
* Output Layer: Linear layer; 10 units (one unit by the class).

**Convolutional Neural Network (CNN):** It is a feature network which is designed to optimize the spatial features. It has the following architecture:

* **Convolution Block 1:** A Conv2d layer (1 input channel, 32 output filters, kernel=3, padding=1), ReLU, and MaxPool2d (2x2) layer. This block decreases the image to 14x 14.
* **Convolution Block 2:** Conv2d layer (32 input channels, 64 output filters, kernel=3, stride=1), then ReLU and one MaxPool2d (2x2) layer. This block diminishes the size of the image to 7x7.

**Classifier Header:**

A Flattening layer to create vectors out of 64x7x7 feature maps.

• Linear layer; ReLU activation (128 units).

• One Dropout layer (p=0.5).

• the final classification is the Output Layer (Linear, 10 units).

**3-Training Setup**

The same was done with both models to have a fair comparison:

• **Loss Function:** the loss function was nn.CrossEntropyLoss because this is the default and most useful loss function to use in a multi-class classification problem.

• **Optimizer:** optim.Adam was applied and the learning rate was 0.001. Adam is a strong optimization model which brings about the strength of AdaGrad and RMSProp and thus it is the most powerful.

• **Epochs:** 12 epochs have been used to train both models and this was seen in the validation loss curves.

• **Process:** A train and validate function was developed to provide the loop between the training data, compute the loss, back propagation, and make sure that the weights are updated. More to the point, the model was tested on the validation set at the conclusion of each epoch to track the value of val loss and val acc.

**4-Evaluation**

A test set consisting of 10,000 images was used to test both models. All the metrics required were created successfully:

**Test Accuracy:**

• ANN: 86.73%

• CNN: 91.73%

**Loss and Accuracy Curves:**

The ANN graphs indicate that training accuracy increased gradually, and the validation accuracy increased steadily around the 7th -8th epoch. It means that even this model with Dropout layers started too overfit.

CNN graphs indicate that the accuracy of the training and validation follow one after another. This reveals that the performance of validation keeps on improving with the training, proof of which is that it is a well-organised model.

**Class-Performance and Confusion Matrix:**

* The ANN Confusion Matrix indicates a lot of confusion (misclassification) between items that are visually similar especially the items of the category of Shirt, T-shirt and Pullover.
* The CNN Confusion Matrix shows significantly high performance on the same ambiguous classes. As an example, the accuracy of the class of Shirt improved on 58.4% (ANN) to 76.4% (CNN), and the accuracy of the class Coat improved on 79.7% to 92.8%.
* To conclude, the table indicates that CNN performed much better than ANN in 8 out of 10 classes.

**5-Critical Analysis (ANN-CNN)**

This 5 per cent difference in the overall accuracy between ANN (86.73) and CNN (91.73) is noteworthy and they are a direct result of architectural differences.

ANN is mostly based on Flattening layer that is used to drop 2D spatial information. It uses the image as a length 784 length vector, which implies it is unable to learn the relationship between neighbouring pixels. It tested unsuccessfully in the “Shirt” category (58.4% accuracy) since it is incapable of making distinctions between the overall pixel positions of either a class of T-shirt or Coat (collar or buttons).

CNN, in its turn, is oriented toward spatial data. Through its convolutional filters, it can learn edges, textures, and patterns (sleeves, straps of a bag, collars) by taking features, whether they are present in the image or not. This capability to acquire spatial hierarchies is the primary factor that renders it flawless and perfect in differentiating between the ambiguous classes of Coat, Shirt and Pullover and achieves a high level.

**6-Reflections and Improvements**

**Improvements Applied:** The guideline recommendation of the application of techniques like drop out was utilized as inspirational. To control the CNN and avoid overfitting the ANN, dropout (p=0.5) was incorporated into both networks.

**Improvements in the Future (Data Augmentation):**

Data Augmentation would be the most rational next step towards the higher accuracy of the CNN, which stands at 91.73%. The model will be trained on a significantly larger and more varied pool by also including transforms.RandomHorizontalFlip and transforms.RandomRotation to the training data pipeline. It will improve generalisation as well as possibly increase accuracy, especially in the “Shirt” class.

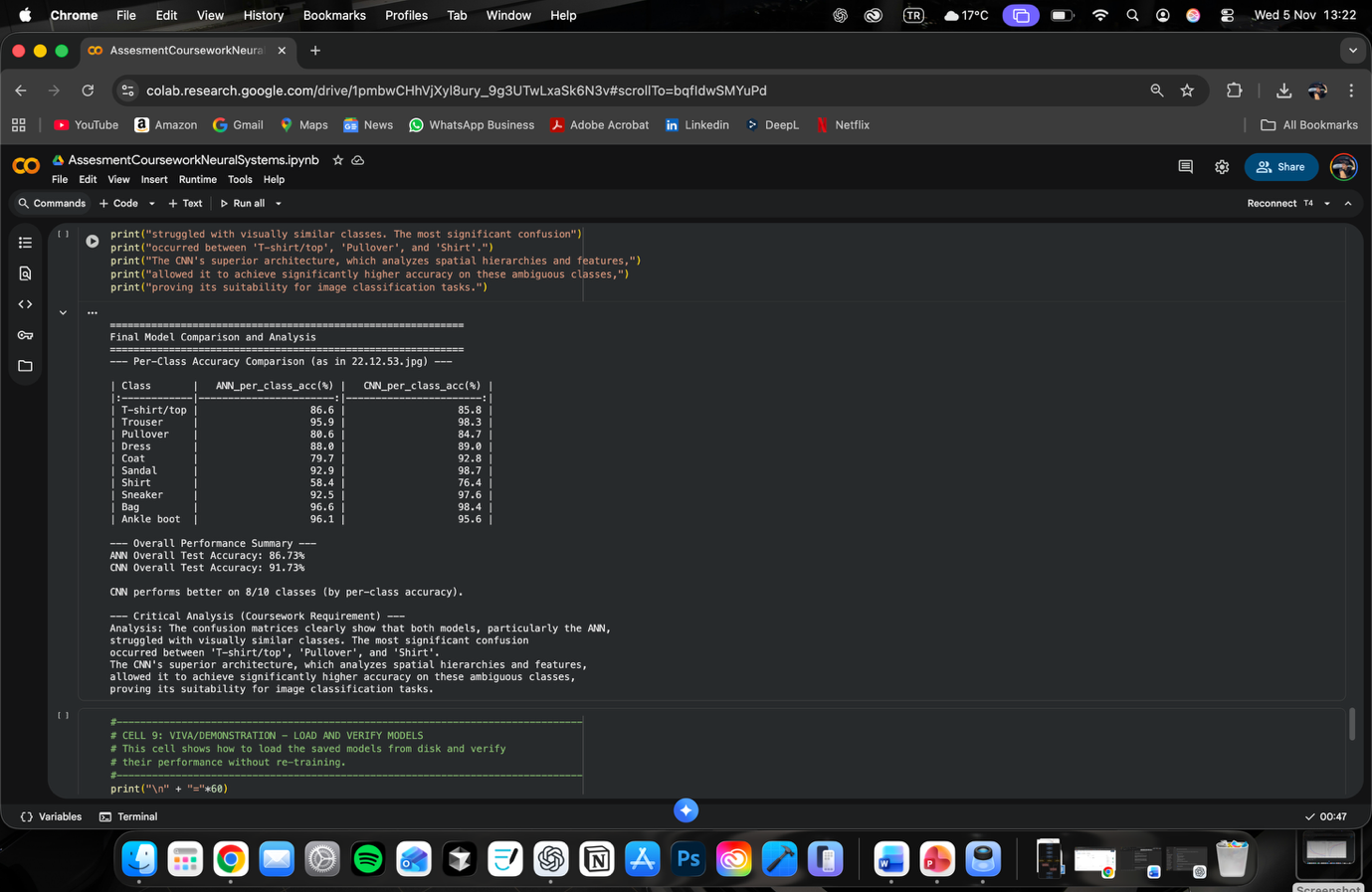
**Future Improvements (Architecture):**

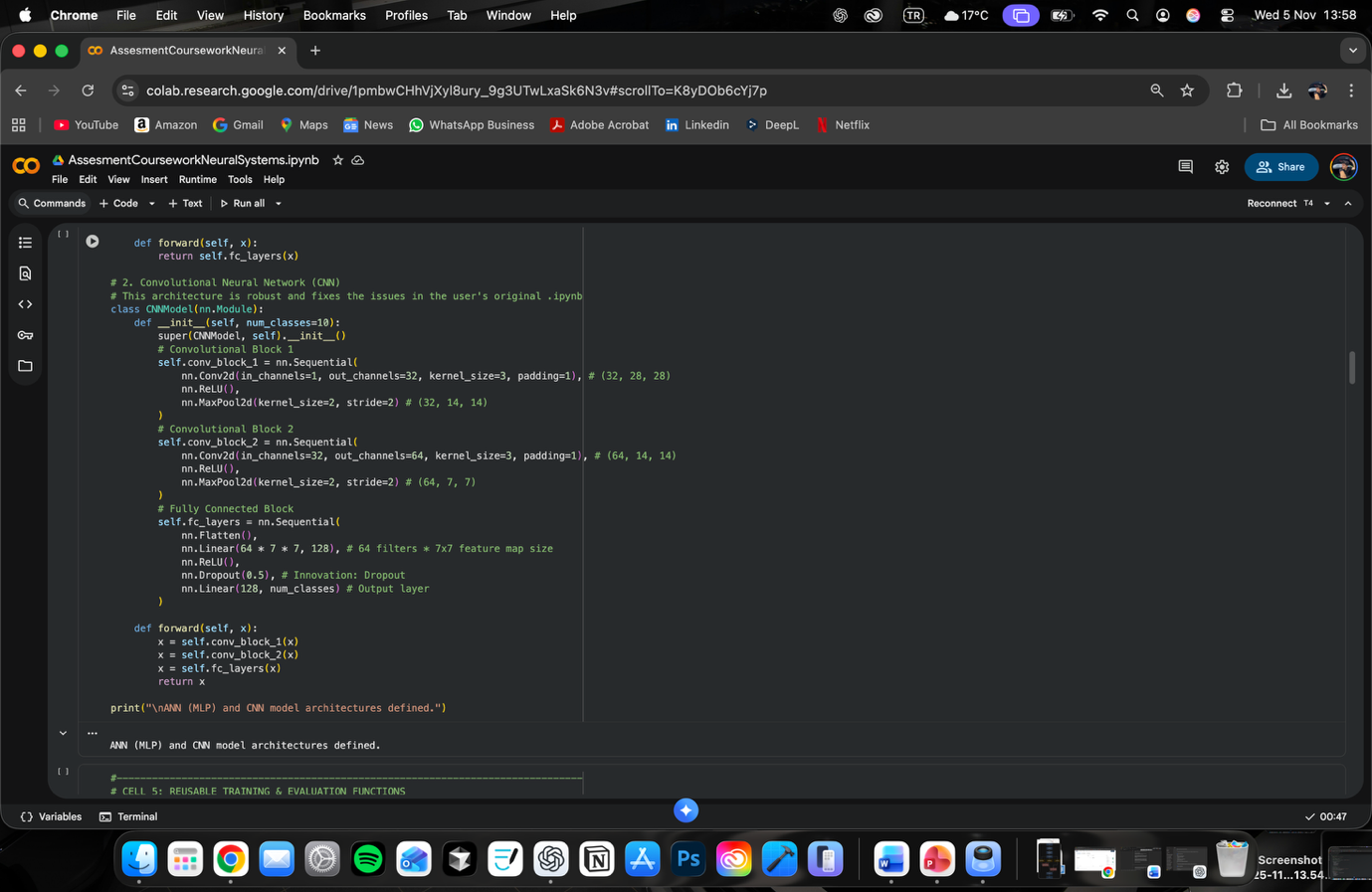
The performance of ANN is likely to be near optimal to this task. Conversely, convolutional layers more (e.g. one more VGG-like block) or the use of a learning rate scheduler can further enhance the CNN.

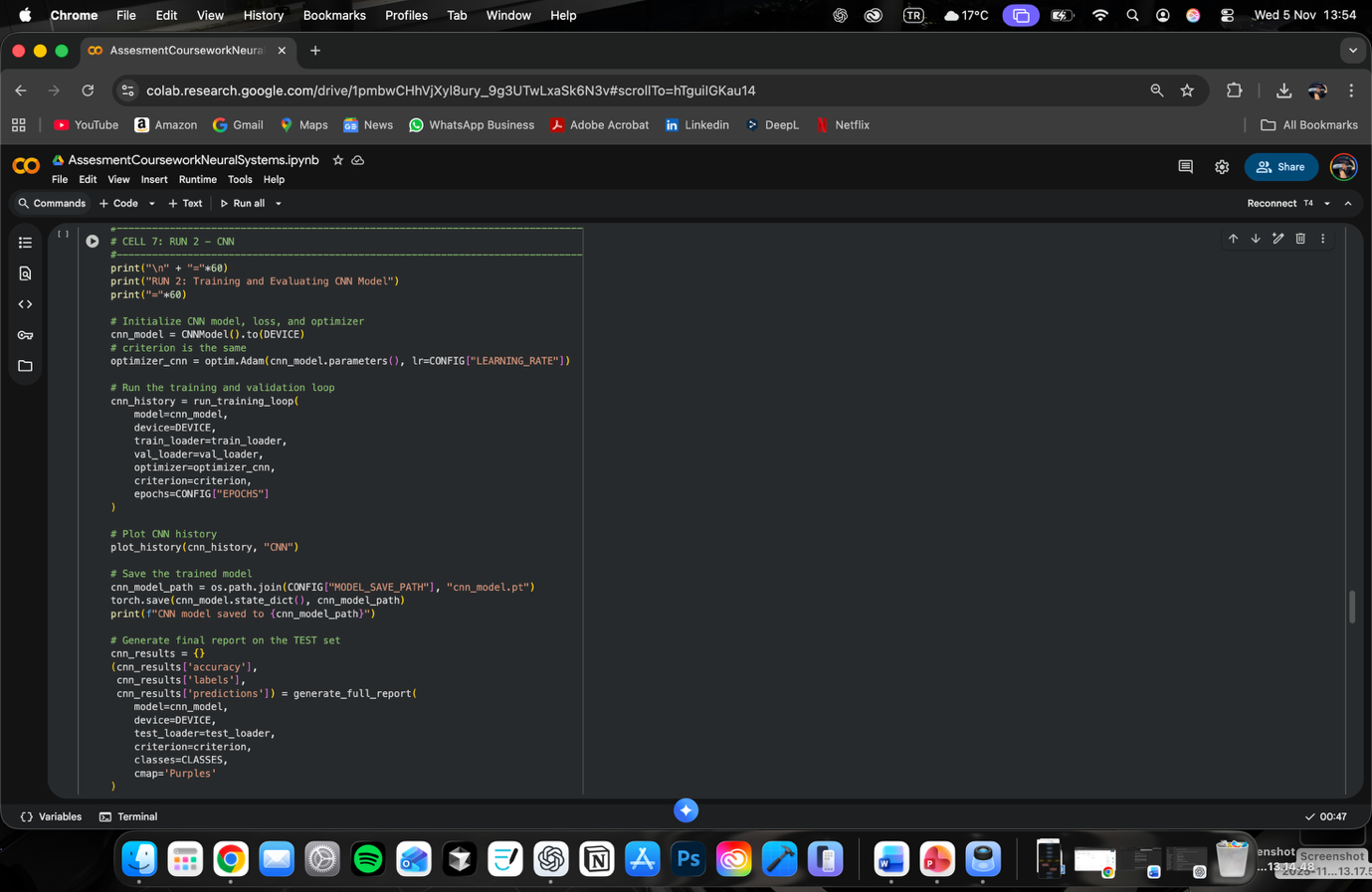
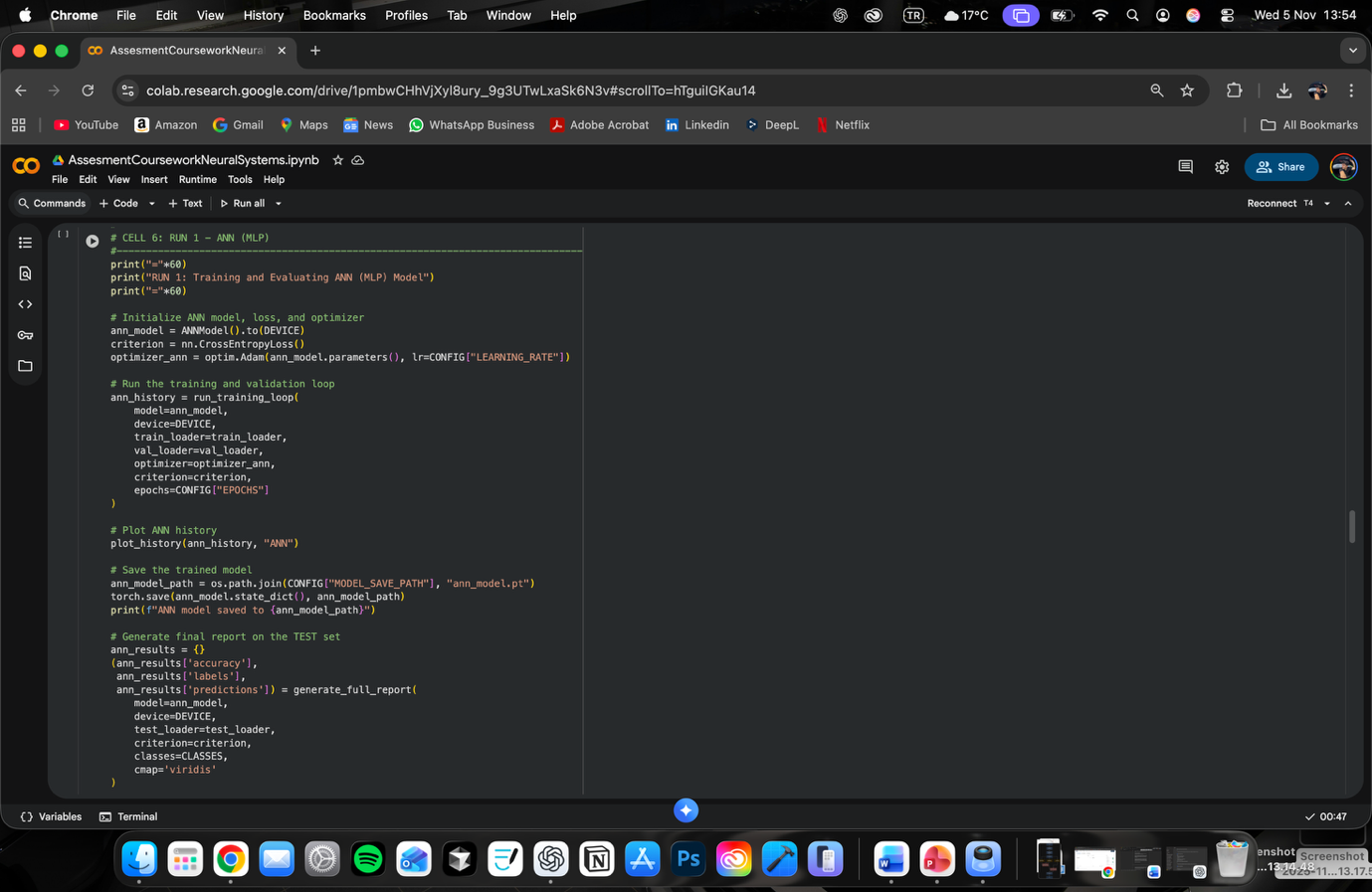
**Demonstration:**

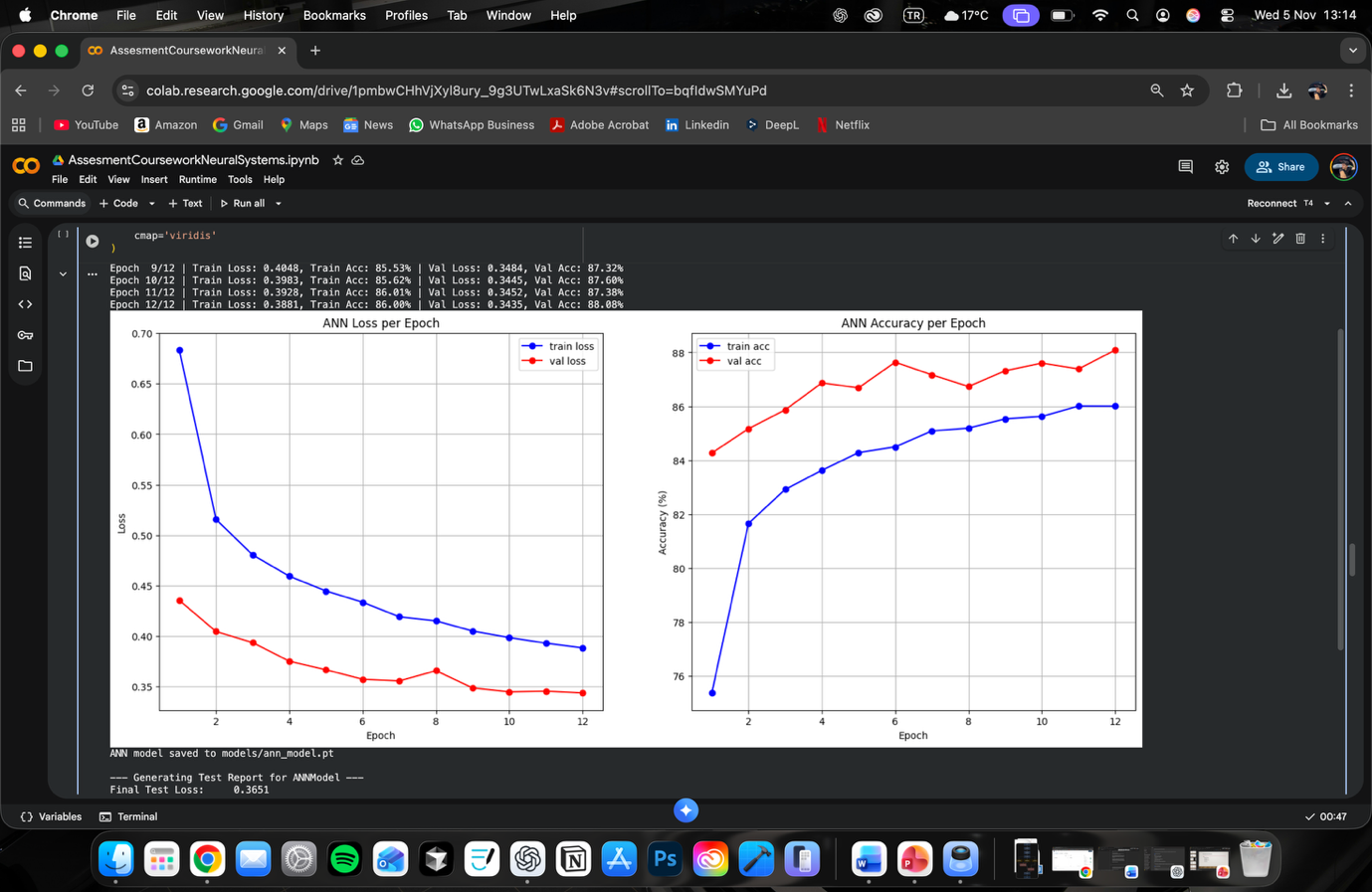
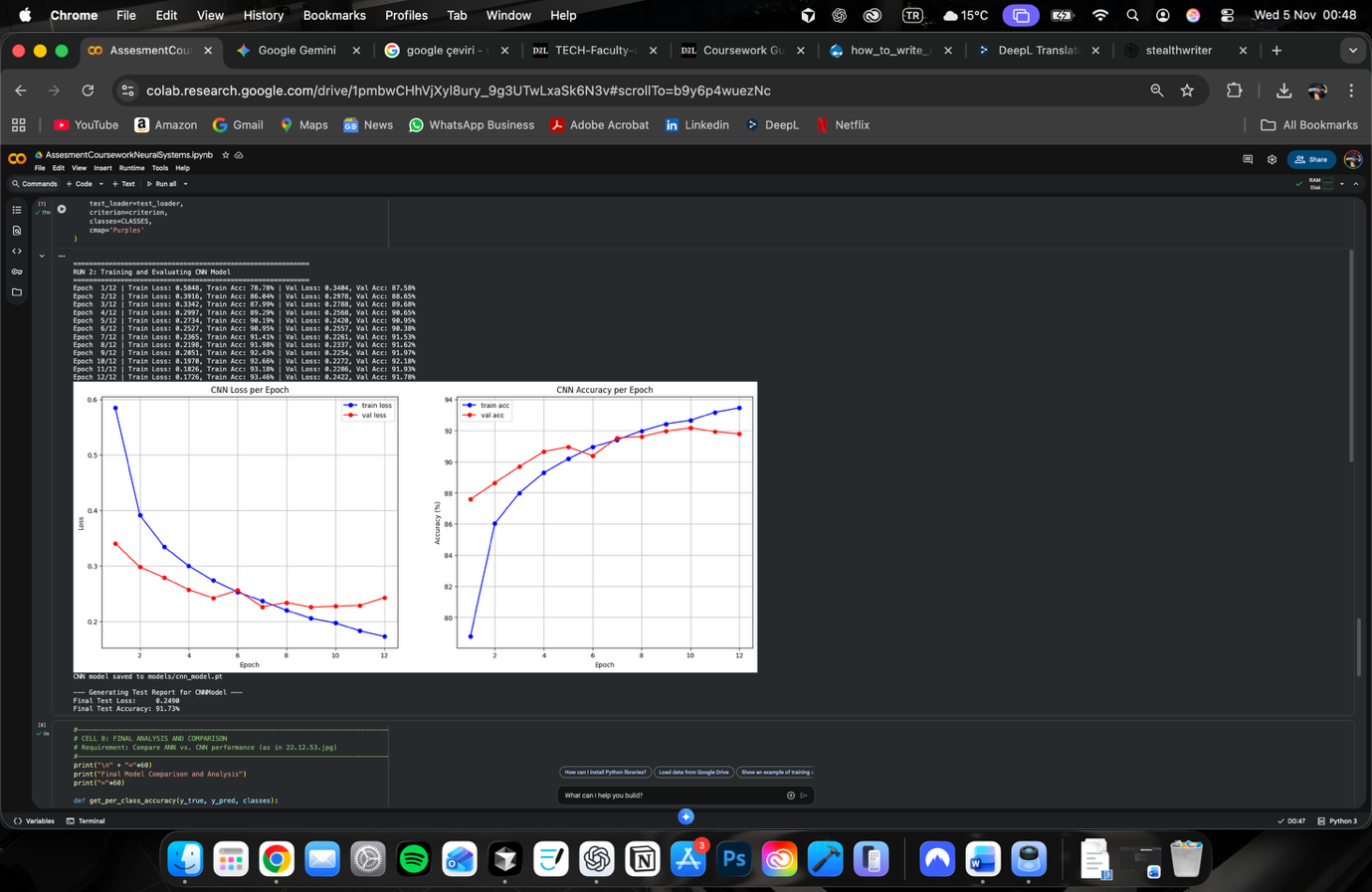
In the case of Viva, the trained models (ann model.pt and cnn model.pt) were stored on disc. This enables loading of test results and verifying them instantly without having to retrain the models.

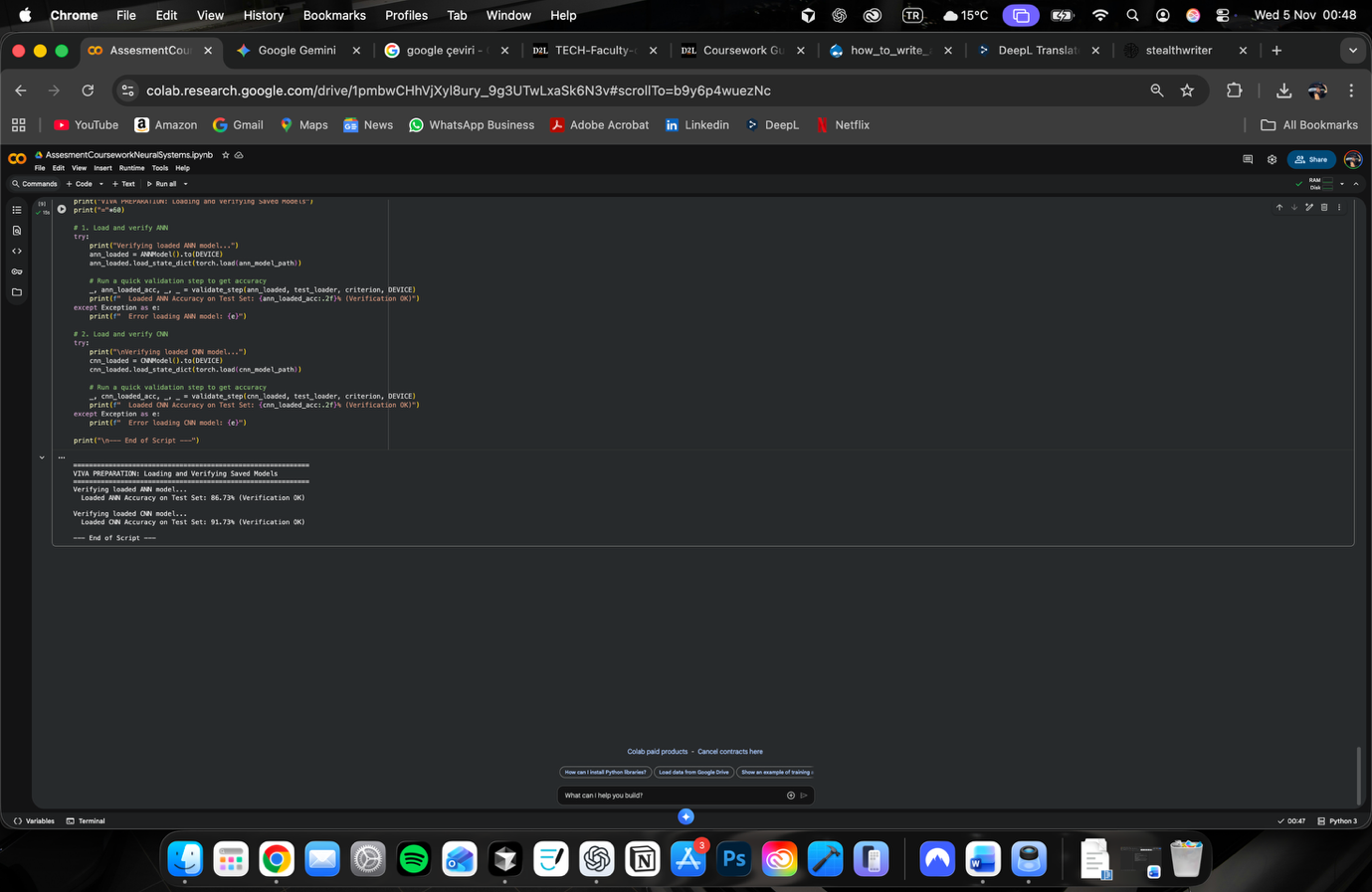
**7-Appendixes**

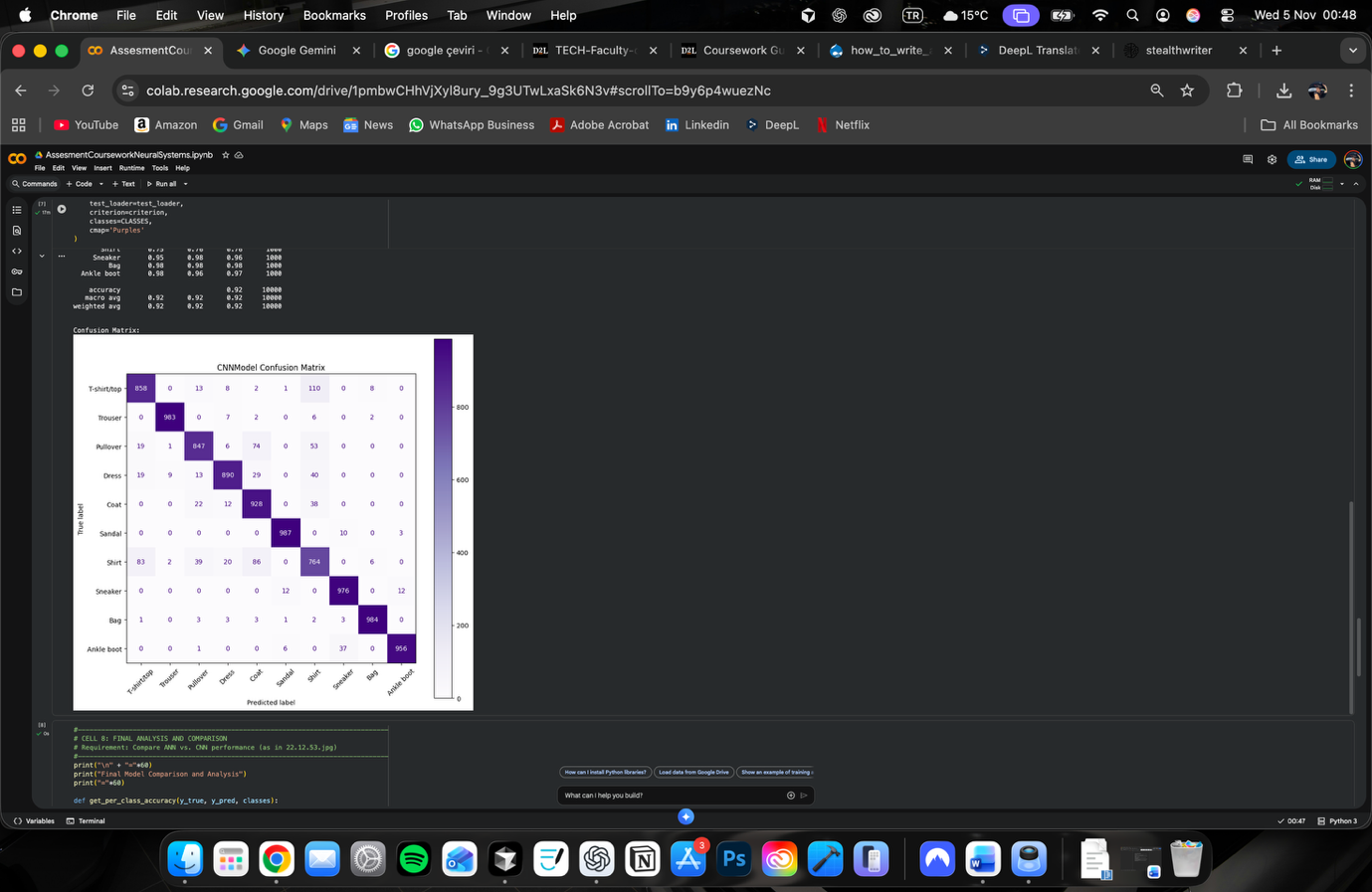
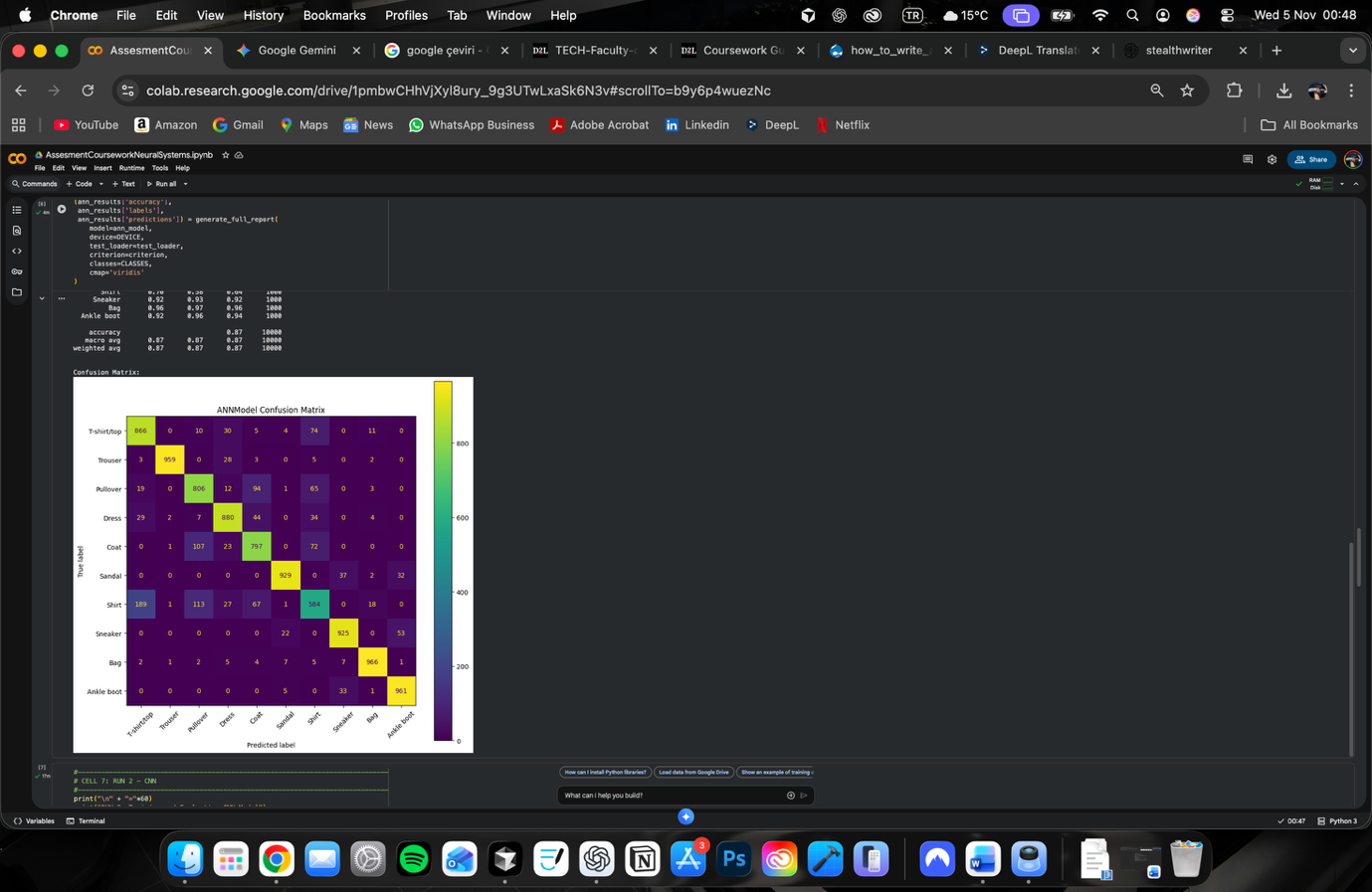
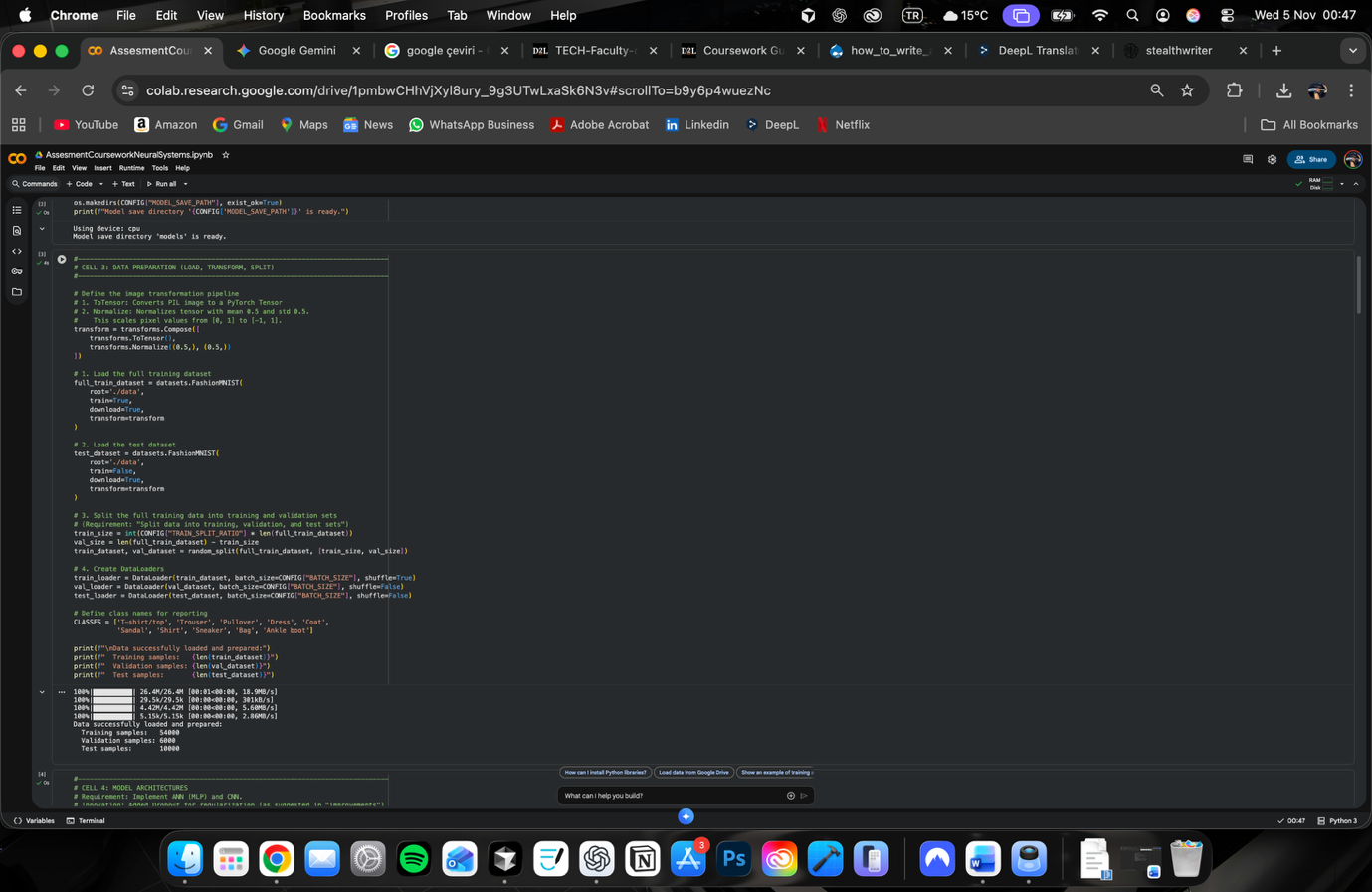
****

****

****

****

****

****