Indira Mallela

Exercise 2: Image Classification using Deep Learning with TensorFlow

Our team approached this exercise to explore and execute Image Classification through TensorFlow as well as Microsoft Azure Custom Vision. I am listing out the process/activities along with screenshots below.

Dataset Selection

We have conducted two experiments using two different datasets to classify images using Tensorflow and two other datasets have been chosen to train, test classification of images using Microsoft Azure Custom Vision. (I am providing snippets of the code for one experiment here and screenshots of execution for all)

Using TensorFlow

- Fashion MNIST dataset (https://keras.io/api/datasets/fashion_mnist/)
  - 10 classifications (T-Shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot)
- MNIST digits classification dataset (https://keras.io/api/datasets/mnist/)
  - Digits zero to nine

```
# Load the Fashion MNIST dataset
(X_train, y_train), (X_test, y_test) =
keras.datasets.fashion_mnist.load_data()
class_names = [
    "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat",
    "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot"
]
```

Using Microsoft Azure Custom Vision

- Simpsons Image Classification
  - Four classifications (Homer, Bart, Lisa, Maggie)
- A subset of Fashion MNIST dataset
  - Four classifications (Bags, Shoes, T-Shirts, Trousers)

Processing of the datasets

Using the below code the images in the dataset are resized and pixels are normalized. Thereupon, the data is split into two sets one for training and another for testing.

```
# Scale the pixel values to a range of 0 to 1
X_train = X_train / 255.0
X_test = X_test / 255.0

# Split the training data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
test_size=0.2, random_state=42)
```

Indira Mallela

Model Selection and Implementation using TensorFlow

Model architecture is defined with four layers i.e. flatten, dense, dropout, and dense for final layout. The optimizer algorithm used to update the weights of the model during training. And "adam" a popular optimization algorithm is used.

```
# Define a function to create the model with adjustable hyperparameters
def create_model(hidden_units=128, dropout_rate=0.2):
    model = keras.Sequential([
        keras.layers.Flatten(input_shape=(28, 28)),
        keras.layers.Dense(hidden_units, activation="relu"),
        keras.layers.Dropout(dropout_rate),
        keras.layers.Dense(10, activation="softmax")
    ])
    model.compile(optimizer="adam", loss="sparse_categorical_crossentropy",
metrics=["accuracy"])
    return model
```

Training and testing the model, and evaluating its performance

Using Keras library a sequential model is created with four layers. Using compile method, we have configured the model for training. “Accuracy” metric is used to select the best model.

```
# Define hyperparameters for tuning
hidden_units = [128, 256]
dropout_rates = [0.2, 0.3]

best_model = None
best_accuracy = 0

# Hyperparameter tuning loop
for units in hidden_units:
    for rate in dropout_rates:
        print(f"\nTraining model with {units} units and {rate} dropout rate")

        # Create and train the model
        model = create_model(hidden_units=units, dropout_rate=rate)
        history = model.fit(X_train, y_train, epochs=10,
validation_data=(X_val, y_val), verbose=2)

        # Evaluate the model on the validation set
        val_loss, val_accuracy = model.evaluate(X_val, y_val, verbose=0)
        print(f"Validation Accuracy: {val_accuracy}")

        # Update the best model if the current model is better
        if val_accuracy > best_accuracy:
```

```
            best_accuracy = val_accuracy
            best_model = model

# Evaluate the best model on the test set
test_loss, test_accuracy = best_model.evaluate(X_test, y_test, verbose=2)
print(f"\nBest Model - Test Loss: {test_loss}, Test Accuracy:
{test_accuracy}")

# Generate predictions using the best model
predictions = best_model.predict(X_test)
predicted_labels = np.argmax(predictions, axis=1)
```

Fine-tuning the model to improve performance

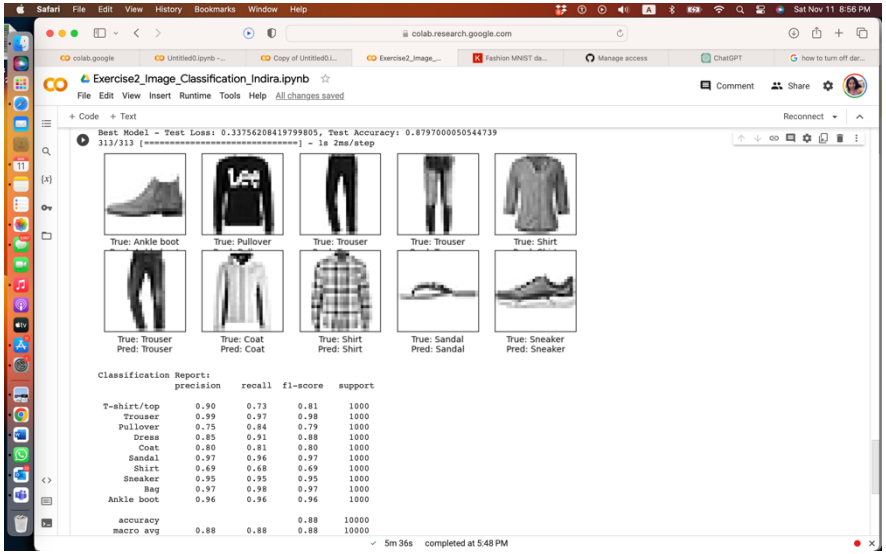
We defined a function to create the neural network model with adjustable hyperparameters. 'Hidden_units' and 'dropout_rates' are used to fine tune the model. Based on highest accuracy best model is identified.

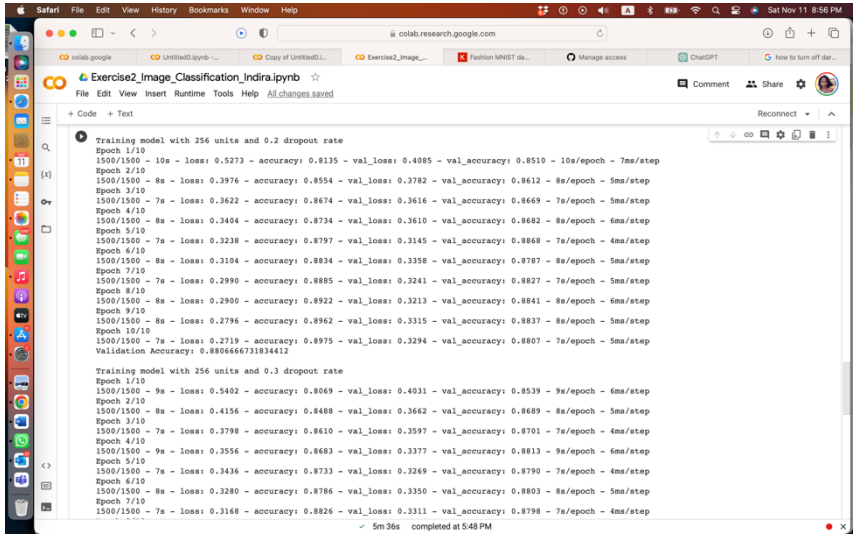
The best model in our model has a test accuracy of 0.8797

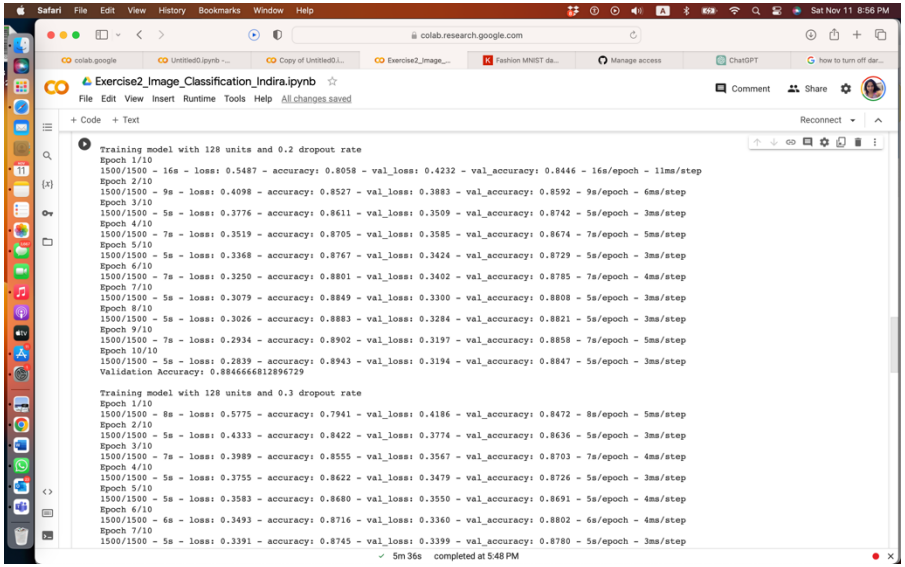
```
# Display the first 10 test images and their predicted labels
plt.figure(figsize=(10, 10))
for i in range(10):
    plt.subplot(5, 5, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(X_test[i], cmap=plt.cm.binary)
    plt.xlabel(f"True: {class_names[y_test[i]]}\nPred:
{class_names[predicted_labels[i]]}")
plt.show()

# Calculate precision and recall
report = classification_report(y_test, predicted_labels,
target_names=class_names)
print("\nClassification Report:\n", report)
```

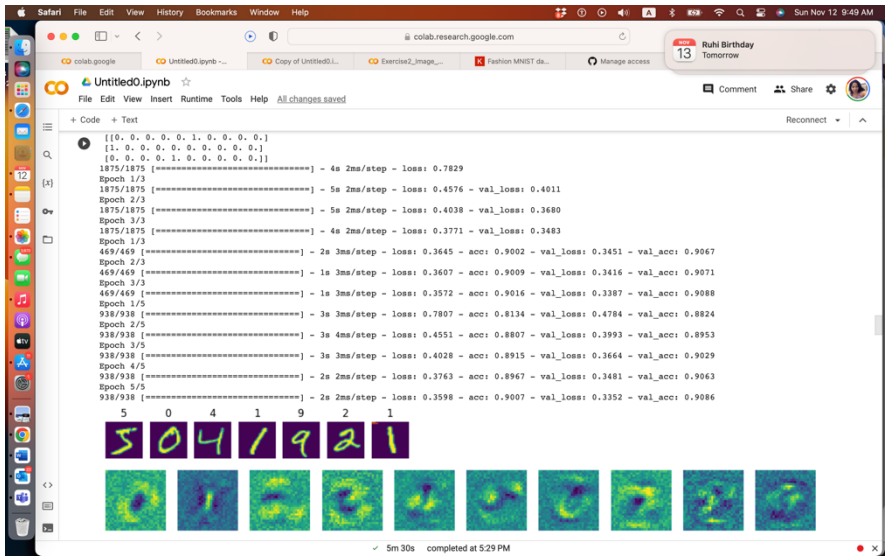
Finally, we generated a list of the first 10 test images and predicted labels to see the results. The next page contains screenshots of execution.

Indira Mallela









Indira Mallela

Training and testing using Microsoft Azure Custom Vision

As a separate experiment, we used Microsoft Azure Custom Vision image recognition service to build, train and test two databases mentioned in the first section of this document. We used 'Custom Vision' web portal. A set of images are downloaded and classified with labels. We used 10% of the images for training Simpsons characters (100/10000). We used close to 80% of the images for training fashion apparel (20/25 for each category). Once images are ready, we followed the steps as below. (Screenshots of execution are in the next pages)

- Create custom vision resources
- Create a new project (classification type)
- Uploaded and tagged training images
- Trained the classifier
- Evaluate the classifier
    - For Simpsons the precision and recall were 84.1% and for fashion apparel and 100% precision and recall for Fashion apparel.

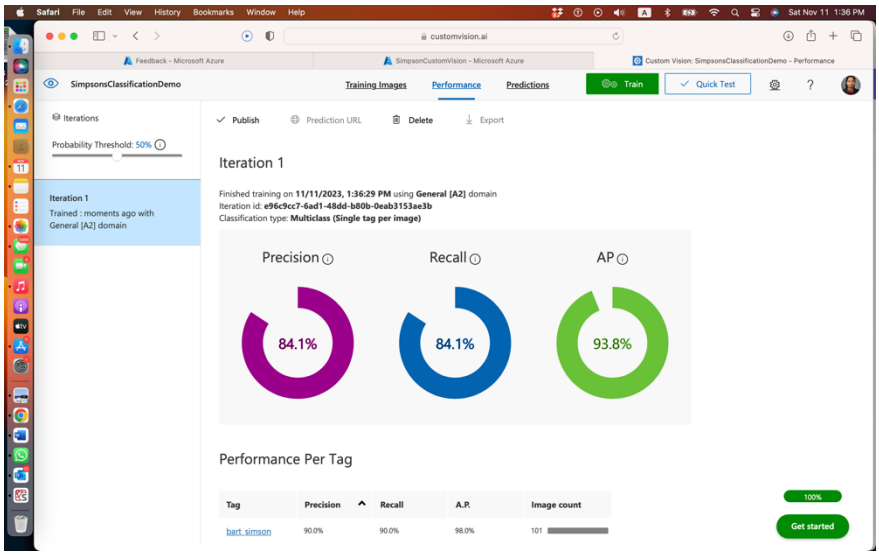
We have used a probability threshold of 50% for classification.

Conclusion and Reflection

This exercise gave me an opportunity to get a very tiny glimpse of how Artificial Intelligence is used in image classification and the enormity of its use and potential scale in the world we live. Microsoft Azure has made the use of these tools simple and easier everyone. We started learning how TensorFlow is used to perform tasks related to training and inference of deep neural networks.

The exercise introduced a new vocabulary for me in terms of Use of neural network, layering the data to perform various transformations on the inputs, probability thresholds, precision, accuracy, recall etc.

To improve accuracy of the models, I would incorporate and explore the concepts of preventing overfitting problems and data quality issues for the said datasets gradually.



Indira Mallela