Team HackWizard

Planck’d 2025

Quantum Computing Hackathon organized by the Quantum Computing Club of IIIT Bangalore, Qimaya.

Meet The Wizards

Quantum Machine Learning Track

Problem Statement-1: MNIST Classification

Dataset: MNIST Dataset

Classical Approach

We built a classical ML model to predict the input images into final class labels using CNN and SVM approach.

1.1 CNN

We constructed a Sequential Convolutional Neural Network (CNN) using **tensorflow.keras**.

The architecture of our model is as follows:

- **Convolutional Block 1:** Conv2D layer with 16 filters (3x3 kernel), followed by a LeakyReLU activation and MaxPooling2D.
- **Convolutional Block 2:** Conv2D layer with 32 filters (5x5 kernel), followed by an ELU activation and MaxPooling2D.
- **Convolutional Block 3:** Conv2D layer with 64 filters (3x3 kernel), followed by an ELU activation and MaxPooling2D.
- **Convolutional Block 4:** Conv2D layer with 128 filters (3x3 kernel) using swish activation, followed by MaxPooling2D.
- **Convolutional Block 5:** Conv2D layer with 256 filters (3x3 kernel) using gelu activation, followed by MaxPooling2D.
- **Fully Connected Layers:** A Flatten layer, followed by a Dense layer of 256 neurons (relu activation), a Dropout of 0.4, a Dense layer of 128 neurons (swish activation), and a Dropout of 0.3.
- **Output Layer:** A Dense layer with 10 neurons and **softmax** activation to produce class probabilities.

The model has a total of **500, 490 trainable parameters**.

Training Methodology

1. **Data Preparation:**
   - The MNIST dataset was loaded.
   - Pixel values were normalized to a [0, 1] range by dividing by 255.0.
   - A channel dimension was added to the images using np.expand_dims.
   - The training and test labels (y_train, y_test) were one-hot encoded using to_categorical.
   - A data_augmentation pipeline (including rotation, translation, zoom, and contrast) was defined, though it was not used in the final model.fit() call .
2. **Compilation:**
   - The model was compiled with the adam optimizer and categorical_crossentropy as the loss function.
3. **Training:**
   - The model was trained using model.fit() for a target of 30 epochs, with a batch size of 64.
   - An EarlyStopping callback was used to monitor val_loss with a patience of 3 epochs.
   - Training stopped early after **Epoch 7**, as the validation loss did not improve sufficiently .

Testing Methodology

1. **Evaluation:** The model's final performance was evaluated on the unseen test set (x_test, y_test) using the model.evaluate() method, which provided the final test loss and test accuracy.
2. **Prediction:** The model.predict() function was called on x_test to get the raw probability predictions for each class.
3. **Label Conversion:** These probabilities were converted into final class labels (0-9) using np.argmax. The one-hot encoded true labels (y_test)

were also converted back to class indices using np.argmax for comparison.

Metrics

You used the following metrics to evaluate your CNN model:

- **Accuracy:** Calculated using both model.evaluate() and accuracy_score from sklearn.metrics.
- **Loss:** Training and validation loss were tracked during training and a final test loss was reported.
- **Classification Report:** A detailed report from sklearn.metrics was generated, showing **precision**, **recall**, and **f1-score** for each digit class.
- **Confusion Matrix:** A confusion matrix was generated and plotted using seaborn.heatmap to visualize correct and incorrect predictions for each class.
- **Training vs. Validation Plots:** You plotted the accuracy and loss curves over epochs for both training and validation sets.

Results

- **Final Test Accuracy: 0.9891** (or 98.91%).
- **Final Test Loss: 0.0493**.
- **Best Validation Accuracy: 0.9921** (achieved during training at Epoch 7).
- **Classification Report:** The final report showed excellent performance, with precision, recall, and f1-scores at or near 0.99 for almost all classes.
- **Confusion Matrix:** The heatmap on page 4 confirms the high accuracy, with very few misclassifications outside the main diagonal.

1.2 SVM (Support Vector Machine)

Our machine learning approach involved using a Support Vector Classifier (SVC) from the **sklearn.svm** library.

We experimented with the model by tuning the regularization parameter **C**, while keeping the kernel consistent. The specific models defined were:

- **Model 1:** SVC(kernel='rbf', C=1.0)
- **Model 2:** SVC(kernel='rbf', C=10)
- **Model 3:** SVC(kernel='rbf', C=50)

1.2.1 Training & Testing Methodology

- **Data Preparation:** The MNIST training (x_train) and test (x_test) datasets, which consist of 28x28 images, were flattened into 1-dimensional vectors of 784 features. This was done using the .reshape(len(...), -1) method.

- **Training:** Each of the three SVM models was trained on the flattened training data (x_train_flat) and its corresponding labels (y_train) using the .fit() method.

- **Testing**: After training, each model was used to generate predictions (y_pred, y_pred_c, y_pred_c1) on the flattened test data (x_test_flat) using the .predict() method.

1.2.2 Metrics

We used several metrics from **sklearn.metrics** to evaluate the performance of our models:

- **Accuracy Score:** accuracy_score was used to get the overall percentage of correct predictions.
- **Confusion Matrix:** confusion_matrix was used to visualize the performance of each model, showing correct and incorrect predictions for each digit class. This was plotted as a heatmap using seaborn.
- **Classification Report:** classification_report was generated for all three models to get a detailed breakdown of precision, recall, and f1-score for each class, along with the overall accuracy.

1.2.3 **Results**

The key results from your SVM model comparison were the test accuracies:

- **Model 1 (C=1.0):**
  - **Test Accuracy:** 0.9792
  - **Classification Report:** Showed an overall accuracy of 0.98 (rounded).
- **Model 2 (C=10):**
  - **Test Accuracy:** 0.9837
  - **Classification Report:** Showed an overall accuracy of 0.98 (rounded).
- **Model 3 (C=50):**
  - **Test Accuracy:** 0.9833
  - **Classification Report:** Showed an overall accuracy of 0.98 (rounded).

Based on these results**, the SVM model with C=10 yielded the highest test accuracy.**