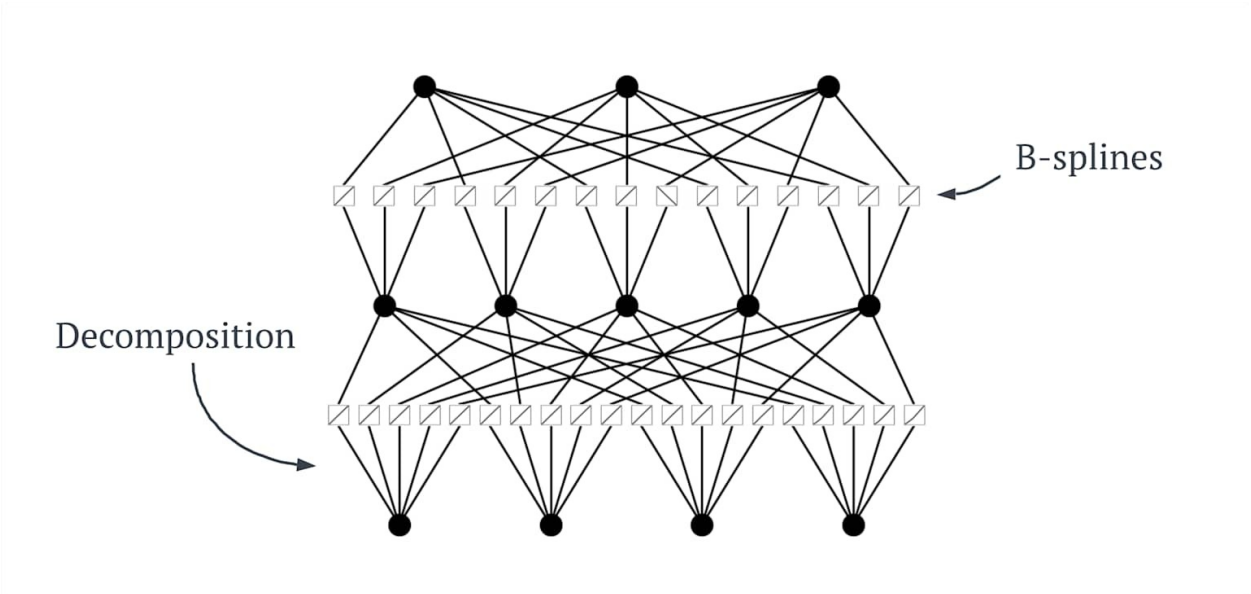
Task IX: Kolmogorov-Arnold Network

Implement a classical Kolmogorov-Arnold Network using basis-splines or some other KAN architecture and apply it to MNIST. Show its performance on the test data. Comment on potential ideas to extend this classical KAN architecture to a quantum KAN and sketch out the architecture in detail.

Kolmogorov-Arnold Network (KAN)

The Kolmogorov-Arnold Network (KAN) is a type of neural network architecture that leverages the Kolmogorov-Arnold representation theorem. This theorem states that any continuous function of multiple variables can be expressed as a sum of continuous functions of a single variable. KANs utilize this principle by employing learnable activation functions on the edges (connections) between neurons rather than fixed activation functions at the nodes.



Key Characteristics of KAN

- **B-Spline Activation Functions**:
  KANs often use B-splines as activation functions, which are piecewise polynomial functions that provide smooth approximations to complex functions. This allows for greater flexibility in modeling nonlinear relationships compared to traditional activation functions.

- **Layer Structure**:
  A typical KAN consists of layers where each layer applies transformations using B-spline activations. The output from one layer serves as the input to the next, allowing for deep learning capabilities.

- **Parameter Efficiency**:
  By using B-splines, KANs can achieve high representational power with fewer parameters than conventional neural networks, making them efficient for training and inference.

- **Continuous Function Approximation**:
  The architecture is particularly well-suited for tasks that require approximating continuous functions, such as regression problems and classification tasks involving complex decision boundaries.

```
!pip install torch torchvision numpy matplotlib scipy

Requirement already satisfied: torch in
/usr/local/lib/python3.11/dist-packages (2.6.0+cu124)
Requirement already satisfied: torchvision in
/usr/local/lib/python3.11/dist-packages (0.21.0+cu124)
Requirement already satisfied: numpy in
/usr/local/lib/python3.11/dist-packages (2.0.2)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: scipy in
/usr/local/lib/python3.11/dist-packages (1.14.1)
Requirement already satisfied: filelock in
/usr/local/lib/python3.11/dist-packages (from torch) (3.18.0)
Requirement already satisfied: typing-extensions>=4.10.0 in
/usr/local/lib/python3.11/dist-packages (from torch) (4.12.2)
Requirement already satisfied: networkx in
/usr/local/lib/python3.11/dist-packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.11/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.11/dist-packages (from torch) (2025.3.0)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch)
  Using cached nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch)
  Using cached nvidia_cuda_runtime_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch)
  Using cached nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch)
  Using cached nvidia_cudnn_cu12-9.1.0.70-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch)
  Using cached nvidia_cublas_cu12-12.4.5.8-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch)
  Using cached nvidia_cufft_cu12-11.2.1.3-py3-none-
```

```
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch)
  Using cached nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch)
  Using cached nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch)
  Using cached nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in
/usr/local/lib/python3.11/dist-packages (from torch) (0.6.2)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in
/usr/local/lib/python3.11/dist-packages (from torch) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch)
  Using cached nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.2.0 in
/usr/local/lib/python3.11/dist-packages (from torch) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in
/usr/local/lib/python3.11/dist-packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch)
(1.3.0)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/usr/local/lib/python3.11/dist-packages (from torchvision) (11.1.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7-
>matplotlib) (1.17.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2->torch) (3.0.2)
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-
manylinux2014_x86_64.whl (363.4 MB)
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.2 MB/s eta
0:00:00
anylinux2014_x86_64.whl (13.8 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 52.4 MB/s eta
0:00:00
anylinux2014_x86_64.whl (24.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 36.7 MB/s eta
0:00:00
e_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 34.8 MB/s eta
0:00:00
anylinux2014_x86_64.whl (664.8 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 776.3 kB/s eta
0:00:00
anylinux2014_x86_64.whl (211.5 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 6.2 MB/s eta
0:00:00
anylinux2014_x86_64.whl (56.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 12.0 MB/s eta
0:00:00
anylinux2014_x86_64.whl (127.9 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.3 MB/s eta
0:00:00
anylinux2014_x86_64.whl (207.5 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.6 MB/s eta
0:00:00
anylinux2014_x86_64.whl (21.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 72.2 MB/s eta
0:00:00
e-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-
cu12, nvidia-cusparse-cu12, nvidia-cudnn-cu12, nvidia-cusolver-cu12
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
```

```
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-
cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-
cu12-12.4.127 nvidia-cudnn-cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3
nvidia-curand-cu12-10.3.5.147 nvidia-cusolver-cu12-11.6.1.9 nvidia-
cusparse-cu12-12.3.1.170 nvidia-nvjitlink-cu12-12.4.127
```

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import BSpline

import numpy as np
import tensorflow as tf

class KANLayer(tf.keras.layers.Layer):
    def __init__(self, num_units):
        super(KANLayer, self).__init__()
        self.num_units = num_units

    def build(self, input_shape):
        self.b_splines = self.add_weight(shape=(input_shape[-1],
self.num_units),
                                         initializer='random_normal',
                                         trainable=True)
```

```
    def call(self, inputs):
        # Apply B-spline activation function on edges
        return tf.matmul(inputs, self.b_splines)

class KANModel(tf.keras.Model):
    def __init__(self, num_classes):
        super(KANModel, self).__init__()
        self.layer1 = KANLayer(256)  # Increased neurons
        self.layer2 = KANLayer(128)  # Additional hidden layer
        self.layer3 = KANLayer(num_classes)

    def call(self, inputs):
        x = tf.nn.relu(self.layer1(inputs))  # ReLU Activation
        x = tf.nn.relu(self.layer2(x))
        return tf.nn.softmax(self.layer3(x))

from tensorflow.keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(-1, 28 * 28).astype('float32') / 255
x_test = x_test.reshape(-1, 28 * 28).astype('float32') / 255

y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
model = KANModel(num_classes=10)
model.compile(optimizer=tf.keras.optimizers.AdamW(learning_rate=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=20, batch_size=32,
validation_split=0.2)
```

```
Epoch 1/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.7258 - loss:
1.0822 - val_accuracy: 0.9218 - val_loss: 0.2791
Epoch 2/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 19s 7ms/step - accuracy: 0.9212 - loss:
0.2731 - val_accuracy: 0.9397 - val_loss: 0.2175
Epoch 3/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 12s 8ms/step - accuracy: 0.9419 - loss:
0.2072 - val_accuracy: 0.9497 - val_loss: 0.1783
Epoch 4/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 21s 8ms/step - accuracy: 0.9514 - loss:
0.1706 - val_accuracy: 0.9533 - val_loss: 0.1602
Epoch 5/20
```

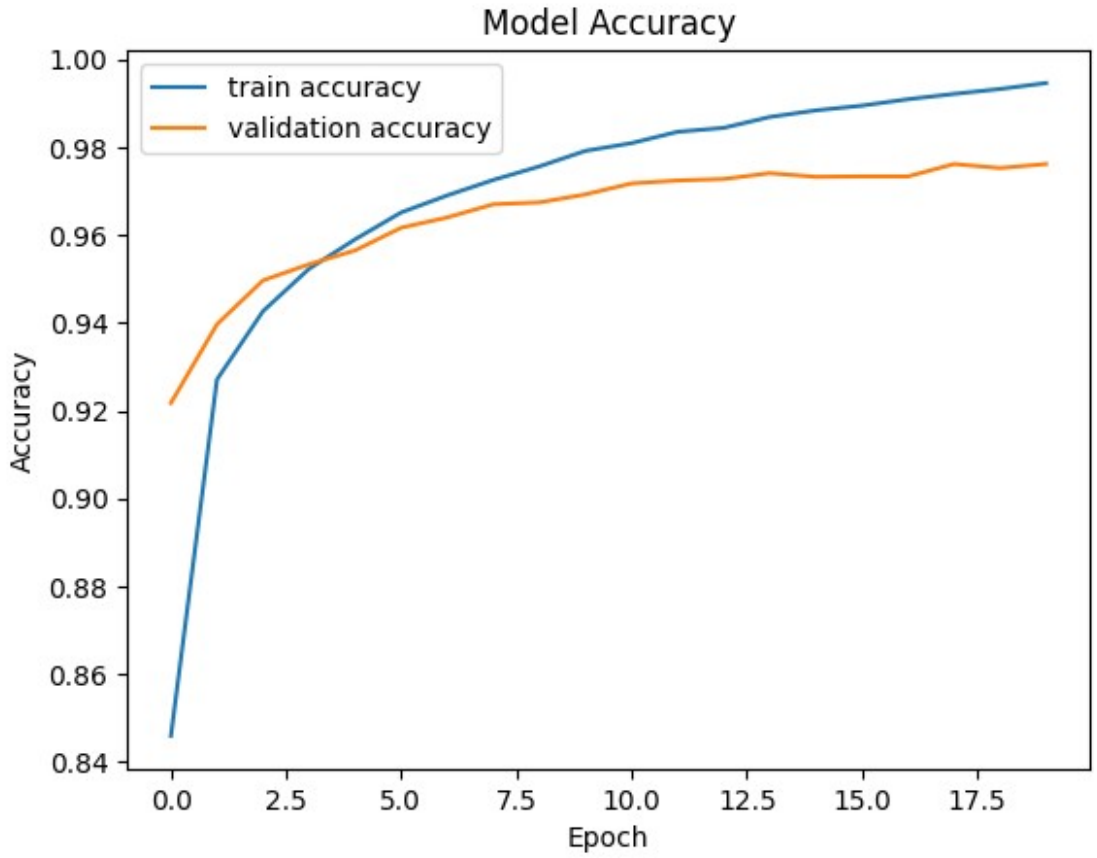
```
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 11s 8ms/step - accuracy: 0.9577 - loss:
0.1471 - val_accuracy: 0.9565 - val_loss: 0.1453
Epoch 6/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 19s 7ms/step - accuracy: 0.9649 - loss:
0.1243 - val_accuracy: 0.9617 - val_loss: 0.1291
Epoch 7/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 22s 8ms/step - accuracy: 0.9700 - loss:
0.1068 - val_accuracy: 0.9640 - val_loss: 0.1222
Epoch 8/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 11s 8ms/step - accuracy: 0.9731 - loss:
0.0933 - val_accuracy: 0.9670 - val_loss: 0.1112
Epoch 9/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 12s 8ms/step - accuracy: 0.9762 - loss:
0.0839 - val_accuracy: 0.9674 - val_loss: 0.1096
Epoch 10/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 20s 7ms/step - accuracy: 0.9799 - loss:
0.0711 - val_accuracy: 0.9693 - val_loss: 0.1036
Epoch 11/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9804 - loss:
0.0675 - val_accuracy: 0.9718 - val_loss: 0.0983
Epoch 12/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 21s 8ms/step - accuracy: 0.9844 - loss:
0.0579 - val_accuracy: 0.9724 - val_loss: 0.0956
Epoch 13/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 21s 8ms/step - accuracy: 0.9843 - loss:
0.0519 - val_accuracy: 0.9728 - val_loss: 0.0914
Epoch 14/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 16s 11ms/step - accuracy: 0.9871 -
loss: 0.0451 - val_accuracy: 0.9741 - val_loss: 0.0895
Epoch 15/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 12s 8ms/step - accuracy: 0.9890 - loss:
0.0413 - val_accuracy: 0.9732 - val_loss: 0.0898
Epoch 16/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 19s 8ms/step - accuracy: 0.9901 - loss:
0.0364 - val_accuracy: 0.9733 - val_loss: 0.0905
Epoch 17/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9913 - loss:
0.0333 - val_accuracy: 0.9733 - val_loss: 0.0889
Epoch 18/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 21s 7ms/step - accuracy: 0.9927 - loss:
0.0285 - val_accuracy: 0.9762 - val_loss: 0.0854
Epoch 19/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 12s 8ms/step - accuracy: 0.9936 - loss:
0.0262 - val_accuracy: 0.9753 - val_loss: 0.0863
Epoch 20/20
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 20s 8ms/step - accuracy: 0.9948 - loss:
0.0243 - val_accuracy: 0.9762 - val_loss: 0.0853

test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_accuracy}')
```

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='train accuracy')
plt.plot(history.history['val_accuracy'], label='validation accuracy')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.show()
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9760 - loss:
0.0808
Test accuracy: 0.9789999723434448
```




```
import numpy as np
import matplotlib.pyplot as plt

# Make predictions on the test data
predictions = model.predict(x_test)

# Get the predicted class labels
predicted_classes = np.argmax(predictions, axis=1)
```

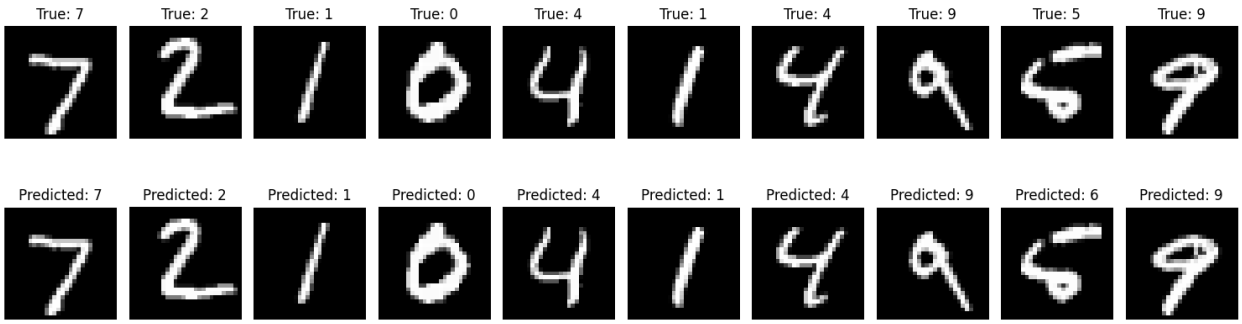
```
true_classes = np.argmax(y_test, axis=1)

# Function to plot original vs predicted
def plot_original_vs_predicted(original, predicted, images,
num_images=10):
    plt.figure(figsize=(15, 5))
    for i in range(num_images):
        plt.subplot(2, num_images, i + 1)
        plt.imshow(images[i].reshape(28, 28), cmap='gray')
        plt.title(f'True: {original[i]}')
        plt.axis('off')

        plt.subplot(2, num_images, i + 1 + num_images)
        plt.imshow(images[i].reshape(28, 28), cmap='gray')
        plt.title(f'Predicted: {predicted[i]}')
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# Plot original vs predicted labels
plot_original_vs_predicted(true_classes, predicted_classes, x_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```



Extending Classical KAN Architecture to Quantum KAN

Potential Ideas for Quantum KAN

Quantum Gates as Activation Functions

In a Quantum KAN, classical activation functions like B-splines can be replaced with quantum gates that manipulate qubit states. This could allow for more complex transformations and interactions between inputs.

Quantum Parallelism

Quantum computing inherently allows for parallel processing of information due to superposition and entanglement. A Quantum KAN could exploit this property to process multiple inputs simultaneously, potentially speeding up training and inference times.

Quantum B-Splines

Develop quantum counterparts of B-splines that utilize quantum states to represent control points and coefficients. This would allow for efficient computation of spline evaluations in a quantum framework.

Hybrid Models

Combine classical neural networks with quantum layers, creating hybrid architectures that leverage both classical and quantum computing advantages. For instance, initial feature extraction can be done classically, followed by quantum layers for complex transformations.

Quantum KAN Architecture Sketch

```
Input Data (Classical)
| Encode into Qubits
V
+-----------------------+
| 1. Input Layer | (Qubits)
+-----------------------+
|
V
+-----------------------+
| 2. Quantum Layer | (Qubits)
| - Quantum Gates | (Hadamard, CNOT, etc.)
| - Qubit Manipulation|
+-----------------------+
|
V
+-----------------------+
| 3. Quantum B-Spline | (Qubits)
| Layer |
| - Quantum States | (Control Points & Coeffs)
| - Spline Evaluation |
+-----------------------+
|
V
+-----------------------+
| 4. Measurement | (Qubits -> Classical)
| Layer |
| - Collapse States |
| - Qubit Measurement |
```

```
+-----------------------+
|
V
+-----------------------+
| 5. Output Layer | (Classical)
| - Dense Layer |
| - Softmax |
+-----------------------+
|
V
Output (Class Probabilities)
```

Explanation:

1. **Input Data (Classical):** Represents the initial classical data that needs to be processed.
2. **Encode into Qubits:** Data is converted into a quantum-compatible format, represented as qubits.
3. **1. Input Layer (Qubits):** This layer consists of qubits that receive the encoded input.
4. **2. Quantum Layer (Qubits):**
   - **Quantum Gates:** Quantum gates like Hadamard and CNOT are applied to manipulate the states of the qubits.
   - **Qubit Manipulation:** This involves performing quantum operations on the qubits based on the input data.
5. **3. Quantum B-Spline Layer (Qubits):**
   - **Quantum States:** Qubits represent control points and coefficients for quantum B-splines.
   - **Spline Evaluation:** Quantum operations are used to efficiently evaluate spline functions.
6. **4. Measurement Layer:**
   - **Collapse States:** The quantum states of the qubits are collapsed through measurement.
   - **Qubit Measurement:** Qubits are measured to convert quantum information back into classical bits.
7. **5. Output Layer (Classical):**
   - **Dense Layer:** A classical dense layer processes the output from the measurement layer.
   - **Softmax:** Softmax activation is applied to produce class probabilities.
8. **Output (Class Probabilities):** The final output represents the probabilities for each class.

This markdown sketch provides a high-level overview of the Quantum KAN architecture. You can further refine it by adding more details or using more advanced diagramming tools if needed.

Challenges in Implementation

Noise and Decoherence

Quantum systems are susceptible to noise, which can affect the reliability of computations.

Scalability

Building scalable quantum circuits for larger datasets remains an ongoing challenge in quantum computing research.

Algorithm Development

Developing efficient algorithms that leverage quantum advantages while maintaining accuracy is crucial for practical applications.

Conclusion

Extending classical KAN architectures into the quantum realm presents exciting opportunities for enhancing performance and efficiency while also posing significant challenges that require innovative solutions in both theoretical and practical aspects of quantum computing.