CS319: Computer Vision

MNIST DIGIT RECOGNITION WITH REAL-TIME WEBCAM INFERENCE

By Gor Mkrtchyan & Inna Krmoyan

MNIST

DATASET OVERVIEW

- MNIST (Modified National Institute of Standards and Technology) Dataset.
- Consists of a collection of 60,000 training and 10,000 testing 28x28 pixel grayscale images of handwritten digits (0 through 9).
- A dataset for image classification.

- Benchmark Dataset: MNIST is a benchmark for evaluating image classification algorithms.
- Digit Recognition Challenge: It involves classifying handwritten digits.
- Gateway to Deep Learning: Used to introduce and implement CNNs.
- Real-world Applications: Fundamental in character recognition and postal code sorting.
- Educational Tool: Common in teaching image classification and neural networks.
- Testing Ground: Researchers test models on MNIST before complex datasets.

The process

1. Loading and exploring the MNIST dataset
2. Data preprocessing
3. Building and training a Convolutional Neural Network (CNN) model
4. Model evaluation and plotting training history
5. Saving the trained model

First & Basic Steps

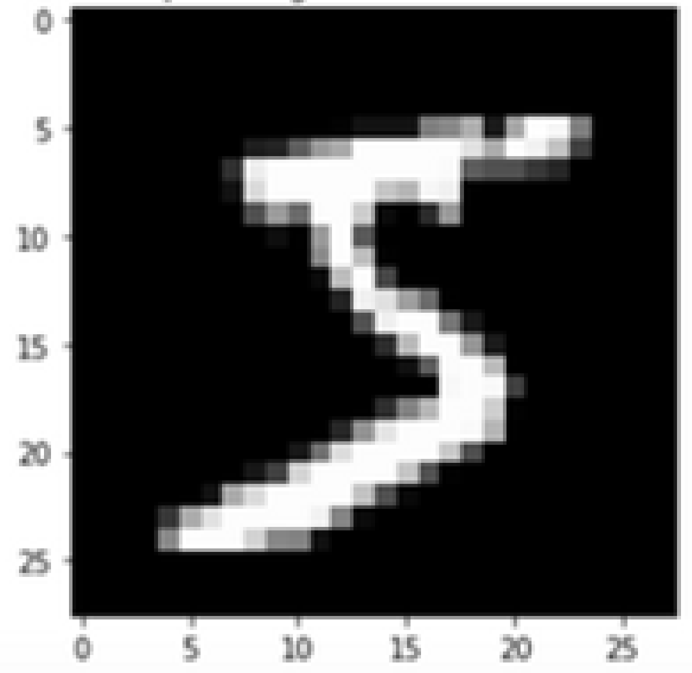
```
# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
# Display information about the dataset
print("Number of training samples:", len(X_train))
print("Number of testing samples:", len(X_test))
print("Shape of an image in the dataset:", X_test[0].shape)
```

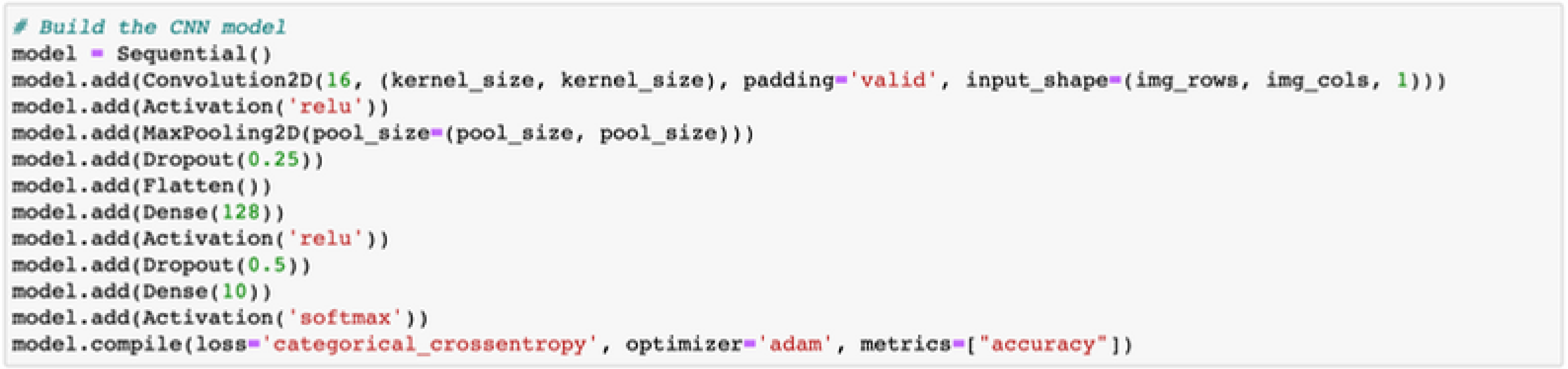
```
Number of training samples: 60000
Number of testing samples: 10000
Shape of an image in the dataset: (28, 28)
```

```
# Display an example image from the training set
plt.imshow(X_train[0], cmap='gray')
plt.title("Example Image from MNIST Dataset")
plt.show()
```

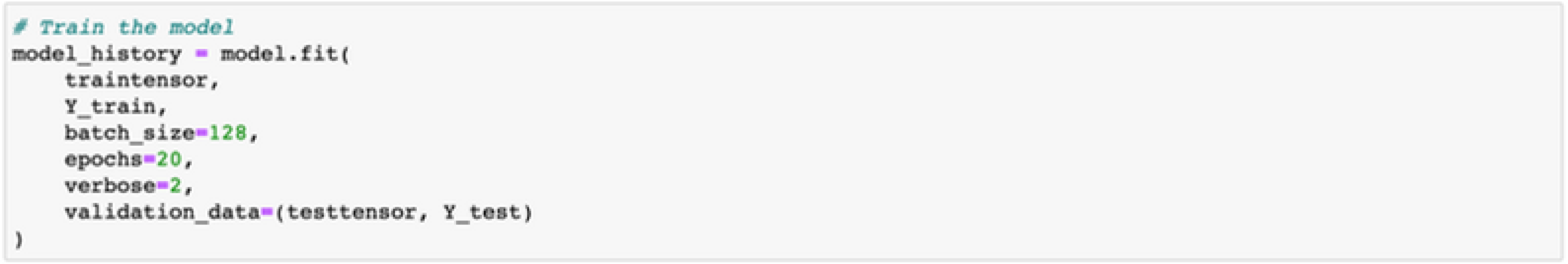
Example Image from MNIST Dataset



After data preprocessing



```
# Build the CNN model
model = Sequential()
model.add(Convolution2D(16, (kernel_size, kernel_size), padding='valid', input_shape=(img_rows, img_cols, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(10))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=["accuracy"])
```

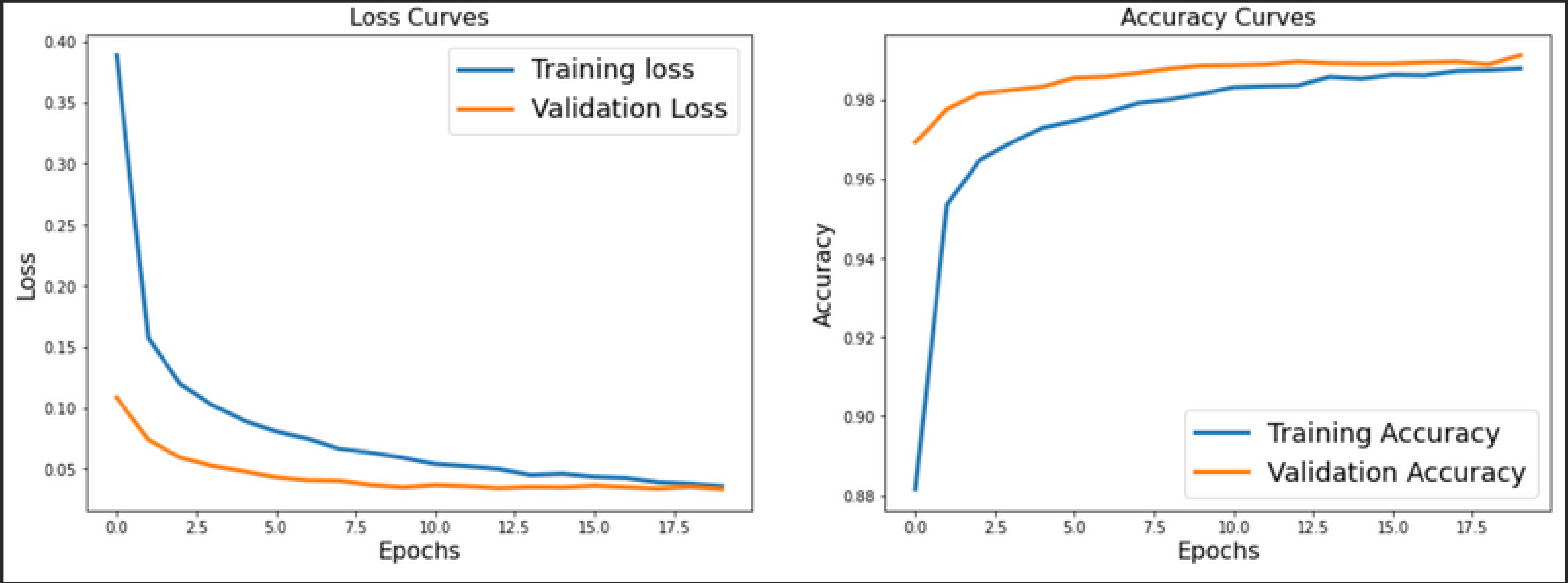


```
# Train the model
model_history = model.fit(
    traintensor,
    Y_train,
    batch_size=128,
    epochs=20,
    verbose=2,
    validation_data=(testtensor, Y_test)
)
```

Model Evalation

Test loss: 0.03362766280770302

Test accuracy: 0.9911000013256073



Real Time Recognition

REAL-TIME INFERENCE

- Loading the pre-trained model
- Accessing the webcam with OpenCV
- Defining a Region of Interest (ROI) in the webcam feed
- Preprocessing frames for digit recognition
- Displaying real-time predictions on the webcam feed

DEMO

THANK YOU!

Do you have any questions?