```
from keras.datasets import mnist
import tensorflow as tf
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from keras.callbacks import ModelCheckpoint
import keras
import numpy as np

# Model construction
model = tf.keras.models.Sequential()
model.add(keras.Input(shape=(28, 28, 1)))

model.add(Conv2D(16, 3, activation = 'relu'))
model.add(MaxPooling2D(2))
model.add(Conv2D(32, 3, activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(100, activation = 'relu'))
model.add(Dense(10, activation = 'softmax'))
print(model.summary())

# Load and transform data
(train_X, train_y), (test_X, test_y) = mnist.load_data()
train_X = train_X.reshape(60000,28,28,1)
train_X = train_X / 255
test_X = test_X.reshape(test_X.shape[0],28,28,1)
test_X = test_X / 255
# Run training
np.random.seed(123)
checkpoint1 = ModelCheckpoint('best_model1.keras',
monitor='val_loss', verbose
                              =1, save_best_only=True)
model.compile('adam', loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x=train_X, y=train_y, epochs = 10,
validation_data=(test_X,test_y),
           callbacks = [checkpoint1])

# Load and transform test data
f = open('/Users/Jensaeh/skola/Neural networks/HW3/part1/
xTest2.bin', 'rb')
canvas_test = f.read()
f.close()
canvas_test = np.frombuffer(canvas_test, dtype='uint8')
canvas_test = canvas_test.reshape(-1, 28, 28, 1)
canvas_test = canvas_test.transpose((0, 2, 1, 3))
canvas_test = canvas_test / 255

# Check data conversion
import matplotlib.pyplot as plt
plt.imshow(canvas_test[0].reshape(28,28), cmap='gray')

# Prediction on test set and canvas test set
```

```
best_model = keras.models.load_model('best_model1.keras')
y_pred_canvas = best_model.predict(canvas_test)
y_pred_canvas = np.argmax(y_pred_canvas, axis = 1)
y_pred = best_model.predict(test_X)
y_pred = np.argmax(y_pred, axis = 1)
from sklearn.metrics import accuracy_score
accuracy_score(test_y, y_pred)

# Save result as CSV
import pandas as pd
pd.DataFrame(y_pred_canvas).to_csv('classifications.csv', index =
False, header
                                    = False)
```