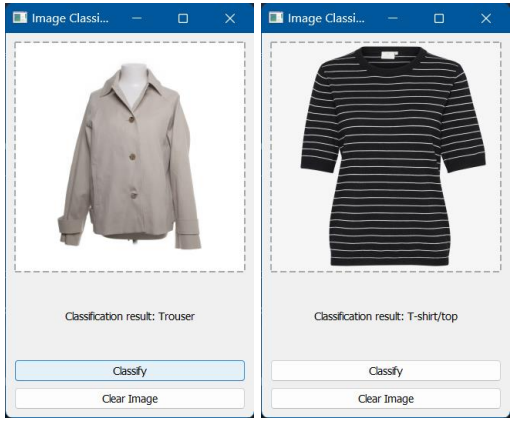
Homework 11

GUI

```python
class MyApp(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Image Classifier for Clothes")
        self.setGeometry(100, 100, 300, 500)

        # create central widget
        c_widget = QWidget(self)
        self.setCentralWidget(c_widget)
        self.mainLayout = QVBoxLayout()
        c_widget.setLayout(self.mainLayout)

        self.image_label = QLabel("Drag an image here or click to select", self)
        self.image_label.setAlignment(Qt.AlignCenter)
        self.image_label.setStyleSheet("border: 2px dashed #aaa;")
        self.image_label.setFixedSize(300, 300)
        self.mainLayout.addWidget(self.image_label)
        self.image_label.mousePressEvent = self.open_file_dialog

        # classification result
        self.result_label = QLabel("Classification result: ", self)
        self.result_label.setAlignment(Qt.AlignCenter)
        self.mainLayout.addWidget(self.result_label)

        # Classify-Button
        self.classify_button = QPushButton("Classify", self)
        self.classify_button.clicked.connect(self.classify_image)
        self.mainLayout.addWidget(self.classify_button)

        # Clear Image-Button
        self.clear_button = QPushButton("Clear Image", self)
        self.clear_button.clicked.connect(self.clear_image)
        self.mainLayout.addWidget(self.clear_button)

        # Load TensorFlow Lite model
        self.interpreter = tflite.Interpreter(model_path="./HW/HW11/fashion_mnist.tflite")
        self.interpreter.allocate_tensors()
        self.input_details = self.interpreter.get_input_details()
        self.output_details = self.interpreter.get_output_details()

        # Enable drag-and-drop functionality
        self.setAcceptDrops(True)
        self.file_path = None
        self.image_loaded = False
```

```python
    def dragEnterEvent(self, event):
        if event.mimeData().hasUrls():
            event.acceptProposedAction()

    def dropEvent(self, event):
        if not self.image_loaded:
            urls = event.mimeData().urls()
            if urls:
                self.file_path = urls[0].toLocalFile()
                self.display_image(self.file_path)

    def open_file_dialog(self, event):
        if not self.image_loaded:
            file_path, _ = QFileDialog.getOpenFileName(self, "Select Image", "", "Images (*.png *.jpg *.jpeg)")
            if file_path:
                self.file_path = file_path
                self.display_image(self.file_path)

    def display_image(self, file_path):
        pixmap = QPixmap(file_path)
        if not pixmap.isNull():
            self.image_label.setPixmap(pixmap.scaled(400, 300, Qt.KeepAspectRatio, Qt.SmoothTransformation))
            self.image_loaded = True
        else:
            self.image_label.setText("Failed to load image")

    def preprocess_image(self, file_path):
        img = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (28, 28))
        img = img.astype(np.float32) / 255.0  # Normalize pixel values
        expected_shape = self.input_details[0]['shape']  # Get expected model input shape
        img = img.reshape(expected_shape)  # reshape dynamically, ensure shape is correct
        return img

    def classify_image(self):
        if self.file_path:
            img = self.preprocess_image(self.file_path)
            if img is None:
                self.result_label.setText("Error loading image")
                return
            output_data = self.interpreter.get_tensor(self.output_details[0]['index'])
            predicted_label = np.argmax(output_data)
            self.result_label.setText(f"Classification result: {class_names[predicted_label]}")
        else:
            self.result_label.setText("No image loaded")
```

```python
    def clear_image(self):
        self.image_label.setText("Original Image")
        self.image_label.setPixmap(QPixmap())
        self.result_label.setText("Classification result: ")
        self.file_path = None
        self.image_loaded = False


if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MyApp()
    window.show()
    sys.exit(app.exec_())
```



Model Architecture

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 8)	80
max_pooling2d (MaxPooling2D)	(None, 14, 14, 8)	0
conv2d_1 (Conv2D)	(None, 14, 14, 16)	1,168
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 16)	0
conv2d_2 (Conv2D)	(None, 7, 7, 32)	4,640
batch_normalization (BatchNormalization)	(None, 7, 7, 32)	128
flatten (Flatten)	(None, 1568)	0
dense (Dense)	(None, 128)	200,832
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

Total params: 208,138 (813.04 KB)
Trainable params: 208,074 (812.79 KB)
Non-trainable params: 64 (256.00 B)

Training/Validation Loss & Accuracy

```
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 18s 10ms/step - acc: 0.7920 - loss: 0.5677 - val_acc: 0.8830 - val_loss: 0.3208
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 15s 10ms/step - acc: 0.8870 - loss: 0.3058 - val_acc: 0.8962 - val_loss: 0.2878
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 15s 10ms/step - acc: 0.9070 - loss: 0.2530 - val_acc: 0.9043 - val_loss: 0.2658
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 14s 9ms/step - acc: 0.9181 - loss: 0.2221 - val_acc: 0.9062 - val_loss: 0.2633
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 14s 10ms/step - acc: 0.9249 - loss: 0.1998 - val_acc: 0.9063 - val_loss: 0.2564
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 14s 10ms/step - acc: 0.9320 - loss: 0.1807 - val_acc: 0.9024 - val_loss: 0.2976
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 14s 9ms/step - acc: 0.9412 - loss: 0.1599 - val_acc: 0.9054 - val_loss: 0.2817
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 14s 9ms/step - acc: 0.9466 - loss: 0.1432 - val_acc: 0.8999 - val_loss: 0.3151
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 14s 9ms/step - acc: 0.9496 - loss: 0.1335 - val_acc: 0.9009 - val_loss: 0.3167
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 13s 9ms/step - acc: 0.9540 - loss: 0.1206 - val_acc: 0.9091 - val_loss: 0.3105
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - acc: 0.8997 - loss: 0.3599
Test Accuracy: 90.17%
```

Name: Josephina Imhoff
Matriculation No.: 23464631
IdM: jo17wila