Understanding the problem:

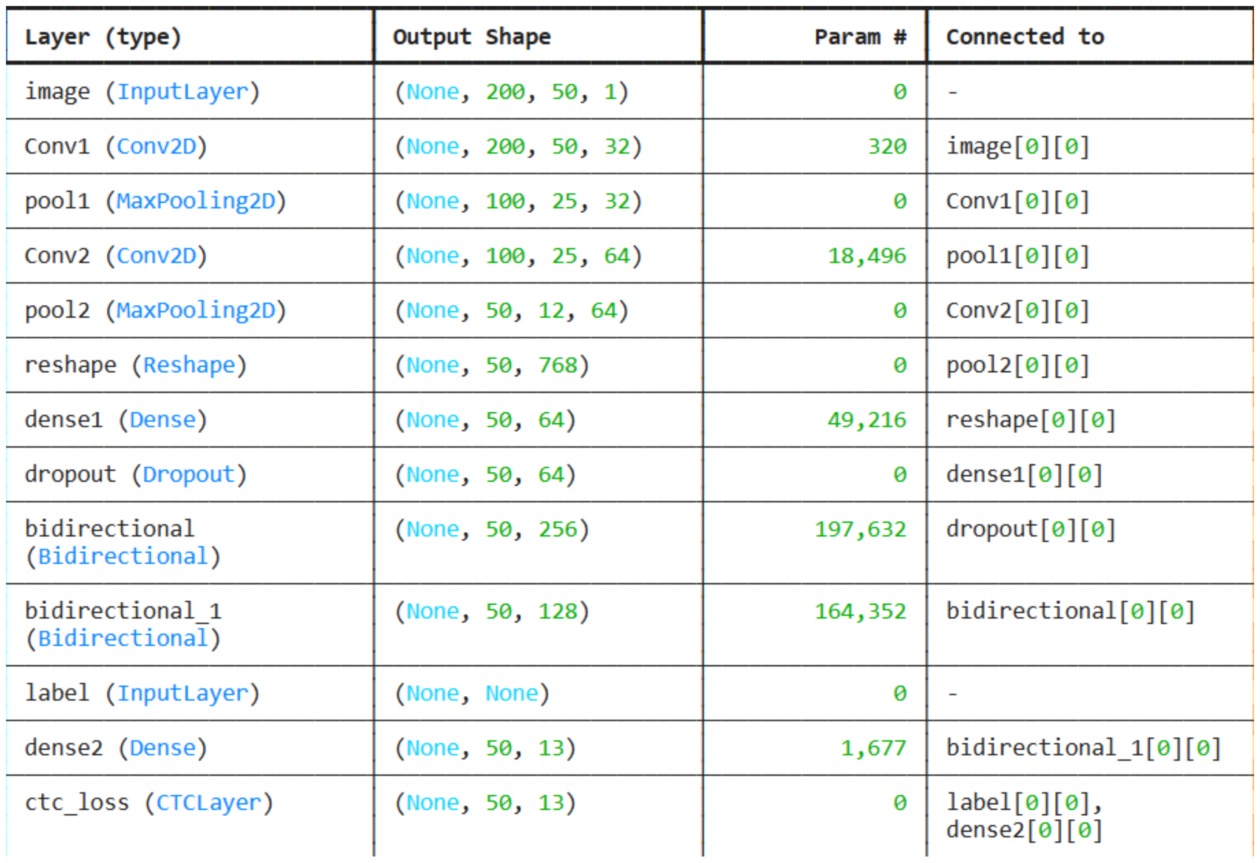
- What we are trying to do is to build an AI system to extract text from images.
- The extracted text can be used for different purposes like filling a form instead of bothering an employee to do it manually.

Data Preprocessing:

- The data contains about 4 images that didn't have a label so I excluded them.
- Resize the images to be (50, 200) height and width respectfully.
- Map the Arabic number to their English equivalent.

Modeling:

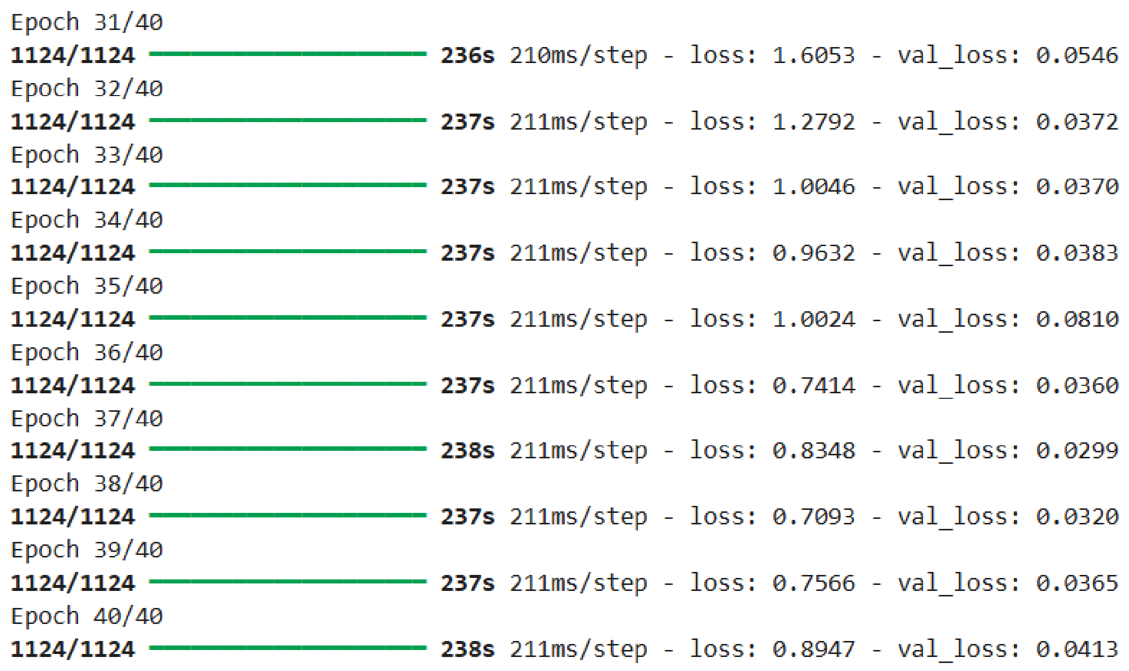
- Use both convolution and LSTM layers to process input images to extract features, transform them into a sequence-like representation.
- Connectionist Temporal Classification (CTC) loss as a loss function, which is commonly used for sequence-to-sequence problems where the alignment between input and output sequences is unknown.
- The model has 431,693 params, they are all trainable.



Layer (type)	Output Shape	Param #	Connected to
image (InputLayer)	(None, 200, 50, 1)	0	-
Conv1 (Conv2D)	(None, 200, 50, 32)	320	image[0][0]
pool1 (MaxPooling2D)	(None, 100, 25, 32)	0	Conv1[0][0]
Conv2 (Conv2D)	(None, 100, 25, 64)	18,496	pool1[0][0]
pool2 (MaxPooling2D)	(None, 50, 12, 64)	0	Conv2[0][0]
reshape (Reshape)	(None, 50, 768)	0	pool2[0][0]
dense1 (Dense)	(None, 50, 64)	49,216	reshape[0][0]
dropout (Dropout)	(None, 50, 64)	0	dense1[0][0]
bidirectional (Bidirectional)	(None, 50, 256)	197,632	dropout[0][0]
bidirectional_1 (Bidirectional)	(None, 50, 128)	164,352	bidirectional[0][0]
label (InputLayer)	(None, None)	0	-
dense2 (Dense)	(None, 50, 13)	1,677	bidirectional_1[0][0]
ctc_loss (CTCLayer)	(None, 50, 13)	0	label[0][0], dense2[0][0]

Training:

- Training the model for 40 epochs.



```
Epoch 31/40
1124/1124 ━━━━━━━━━━━━━━━━━━━━ 236s 210ms/step - loss: 1.6053 - val_loss: 0.0546
Epoch 32/40
1124/1124 ━━━━━━━━━━━━━━━━━━━━ 237s 211ms/step - loss: 1.2792 - val_loss: 0.0372
Epoch 33/40
1124/1124 ━━━━━━━━━━━━━━━━━━━━ 237s 211ms/step - loss: 1.0046 - val_loss: 0.0370
Epoch 34/40
1124/1124 ━━━━━━━━━━━━━━━━━━━━ 237s 211ms/step - loss: 0.9632 - val_loss: 0.0383
Epoch 35/40
1124/1124 ━━━━━━━━━━━━━━━━━━━━ 237s 211ms/step - loss: 1.0024 - val_loss: 0.0810
Epoch 36/40
1124/1124 ━━━━━━━━━━━━━━━━━━━━ 237s 211ms/step - loss: 0.7414 - val_loss: 0.0360
Epoch 37/40
1124/1124 ━━━━━━━━━━━━━━━━━━━━ 238s 211ms/step - loss: 0.8348 - val_loss: 0.0299
Epoch 38/40
1124/1124 ━━━━━━━━━━━━━━━━━━━━ 237s 211ms/step - loss: 0.7093 - val_loss: 0.0320
Epoch 39/40
1124/1124 ━━━━━━━━━━━━━━━━━━━━ 237s 211ms/step - loss: 0.7566 - val_loss: 0.0365
Epoch 40/40
1124/1124 ━━━━━━━━━━━━━━━━━━━━ 238s 211ms/step - loss: 0.8947 - val_loss: 0.0413
```

App:

- Using Flask as a framework to build the app.
- I had some problems using the onnex version of the model so I used h5 one but it needs to be compiled again.
- helper functions to preprocess the images and decode the text in order to show a more readable version of the output.
- Dockerize the app.

```
.
├── main.py        # Python script for your Flask app
├── model.h5        # Pre-trained model for OCR predictions
├── templates/      # Folder containing HTML files for the web UI
│   └── index.html
└── Dockerfile      # Dockerfile for containerizing the application
```