# Misr University For Science & Technology

**College:** Information Technology

## Project by


**Fady Amr 94304**

**Abdelrahman Wael 94109**

**Ahmed Elsayed 94215**

**Eslam Mohammed 94222**

**Muhammad Usamah 94284**


**Course Name: Ai Programming Languages**

**Course Code: AI 301**

## Essay Title: Egyptian Car Plates Reader

**Supervised by:**

Dr:/ Mohammad Tawfik

# Problem Definition

- Currently, we have a problem in our university, where for a car to enter the gates and traverse through the uni grounds, human intervention is required.
- We aim to solve this problem with the powers of Deep Learning and Neural Networks, thus automating the whole process.

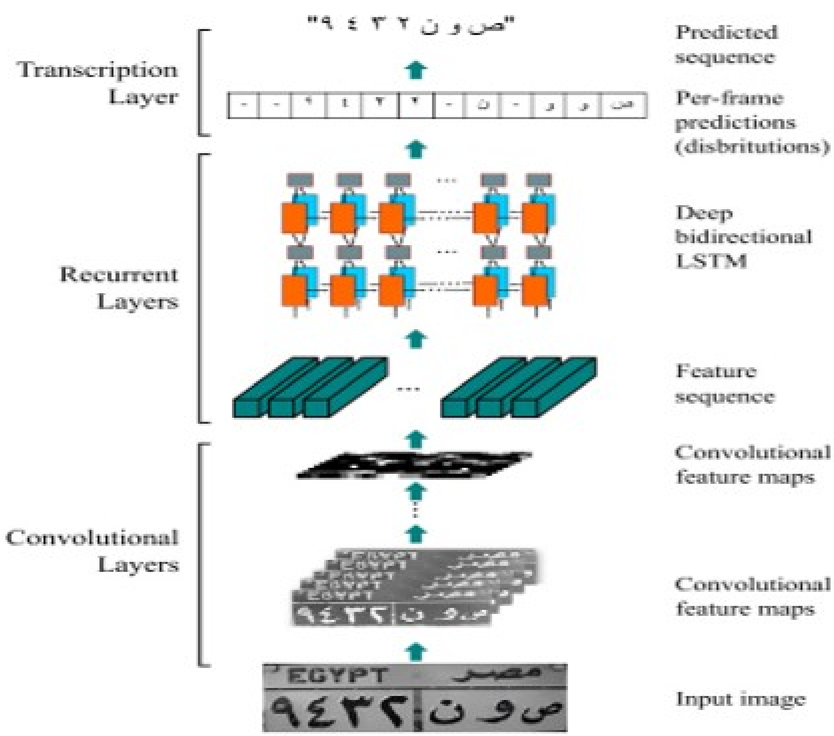
# Proposed Network architecture:

The network architecture. The architecture consists of three parts:

1) The convolutional layers, extract a feature sequence from the input image.

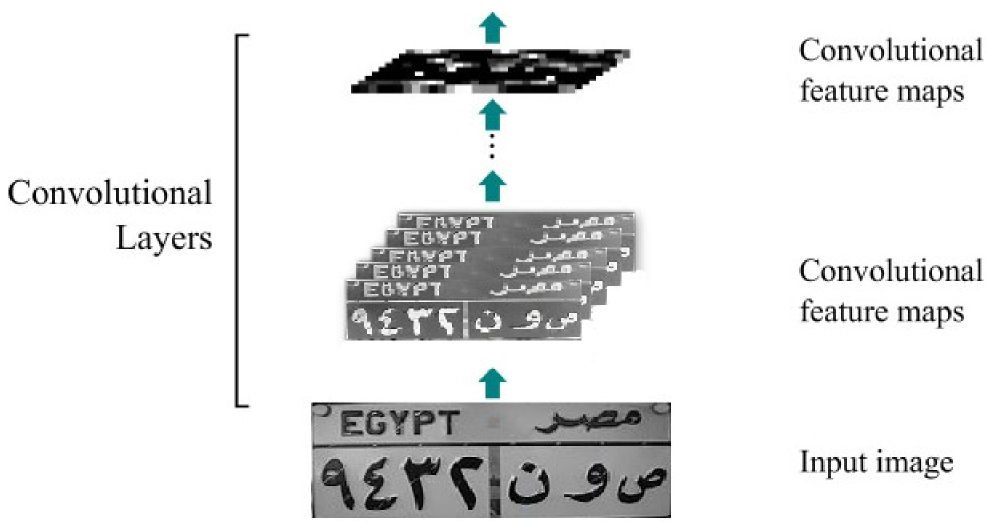
2) The recurrent layers, which predict a label distribution for each frame.

3) The transcription layer, which translates the per-frame predictions into the final label sequence.



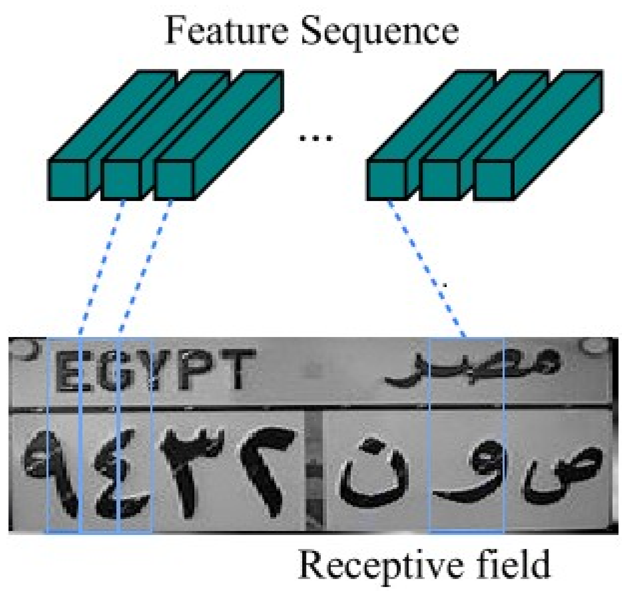
## 1- <u>The convolutional layers:</u>

- The architecture of the convolutional layers is based on the VGG architecture.
- The convolutional layers automatically extract a feature sequence from each input image.



## CNN output to RNN:

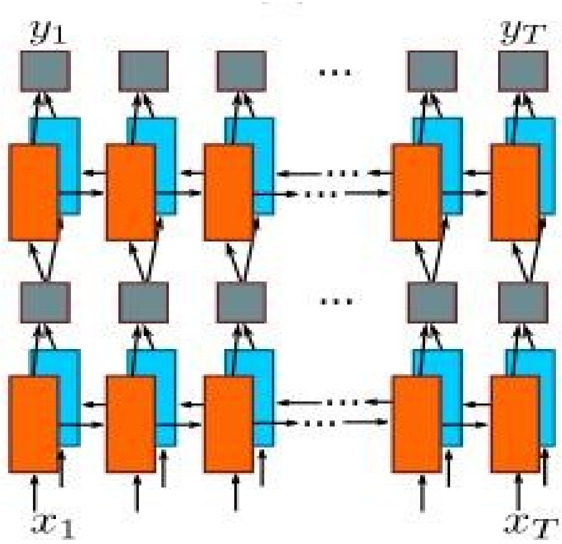
Each vector in the extracted feature sequence is associated with a receptive field on the input image and can be considered the feature vector of that field.



## 2- <u>The Recurrent layers:</u>

A deep bidirectional Recurrent Neural Network is built on the top of the convolutional layers.

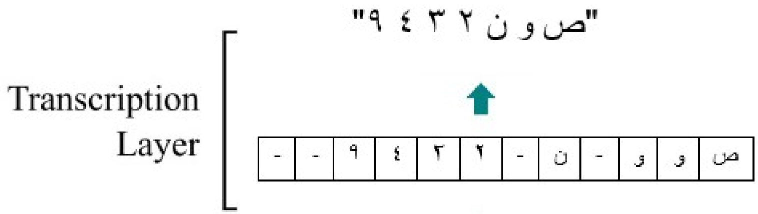
We used bidirectional LSTM




Combining a forward (left to right) and a backward (right to left) LSTMs results in a bidirectional LSTM. Stacking multiple bidirectional LSTM results in a deep bidirectional LSTM.

## 3- The transcription layer:

- Transcription is the process of converting the per-frame predictions made by RNN into a label sequence.
- Mathematically, transcription is to find the label sequence with the highest probability conditioned on the per-frame predictions. In practice, there exist two modes of transcription, namely lexicon-free and lexicon-based transcriptions.
- A lexicon is a set of label sequences that prediction is constrained to, e.g., a spell-checking dictionary.
- In lexicon-free mode, predictions are made without any lexicon. In lexicon-based mode, predictions are made by choosing the label sequence that has the highest probability.
- We used lexicon-free mode.



# Model Summary

```
==================================================================================================
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 the_input (InputLayer)         [(None, 128, 64, 1)  0           []
                                ]

 VGG_Block1 (VggBlock1)         (None, 64, 32, 64)   896         ['the_input[0][0]']

 VGG_Block2 (VggBlock1)         (None, 32, 16, 128)  74368       ['VGG_Block1[0][0]']

 VGG_Block3 (VggBlock2)         (None, 32, 8, 256)   887296      ['VGG_Block2[0][0]']

 VGG_Block4 (VggBlock2)         (None, 32, 4, 512)   3544064     ['VGG_Block3[0][0]']

 VGG_Block5 (VggBlock1)         (None, 32, 4, 512)   2361856     ['VGG_Block4[0][0]']

 reshape (Reshape)              (None, 32, 2048)     0           ['VGG_Block5[0][0]']

 dense1 (Dense)                 (None, 32, 64)       131136      ['reshape[0][0]']

 BI_LSTM_Block1 (BI_LSTM_Block)  (None, 32, 256)     658432      ['dense1[0][0]']

 BI_LSTM_Block2 (BI_LSTM_Block)  (None, 32, 256)     395264      ['BI_LSTM_Block1[0][0]']

 dropout (Dropout)              (None, 32, 256)      0           ['BI_LSTM_Block2[0][0]']

 dense2 (Dense)                 (None, 32, 31)       7967        ['dropout[0][0]']

 softmax (Activation)           (None, 32, 31)       0           ['dense2[0][0]']

 the_labels (InputLayer)        [(None, 7)]          0           []

 input_length (InputLayer)      [(None, 1)]          0           []

 label_length (InputLayer)      [(None, 1)]          0           []

 ctc (Lambda)                   (None, 1)            0           ['softmax[0][0]',
                                                                  'the_labels[0][0]',
                                                                  'input_length[0][0]',
                                                                  'label_length[0][0]']

==================================================================================================
Total params: 8,061,279
Trainable params: 8,055,775
Non-trainable params: 5,504
__________________________________________________________________________________________________
```

## Optimizer:

- For optimization, we use the ADADELTA to automatically calculate per-dimension learning rates. Compared with the conventional momentum method, ADADELTA requires no manual setting of a learning rate. More importantly, we find that optimization using ADADELTA converges faster than the momentum method.

## Loss Function:

For the loss function, we used **Connectionist Temporal Classification (CTC).**

**Why we used CTC?**

- We only have to tell the CTC loss function of the text that occurs in the image. Therefore, we ignore both the position and width of the characters in the image.
- No further processing of the recognized text is needed.

# DATASET:

- Training set = 9947 image
- Images dimensions = (128, 64, 1) → greyscale

## Augmentation Methods:

## Rotation


## Random Erasing


## Random Distortion


## MIX AND MATCH!!

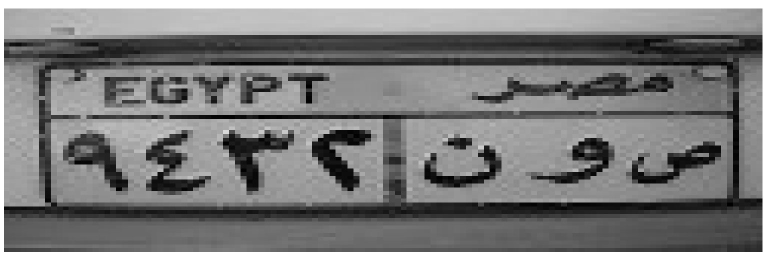
# Data Labeling:

We mapped Arabic letters and numbers to English ones

ص و ن ٢ ٣ ٤ ٩ Will be 9432NWS

If the number of characters in the plate number is less than 7, we use the letter X for padding.

و ن ٢ ٣ ٤ ٩ Will be 9432NWX



# Training the model:

Training Parameters:

- Epochs = 50
- Batch size = 8
- Steps per epoch = dataset size/batch size (9947 / 8 = 1243)

```
history = model.fit(ds_train.next_batch(),
                    steps_per_epoch=int(ds_train.n / batch_size),
                    epochs=50)
```

## System Pipeline:

### Plate Detection:

We used transfer learning to fine-tune an object detection (YOLOv7) model.

### Plate Reading:

Then, it's passed to our model so that it can read the plates.

### Database:

Finally, the plate's numbers are sent to the database to check if it's in it or not.

# <u>**GUI:**</u>