



Ain Shams University

Faculty of Engineering

CESS Program

CSE440: Selected Topics in Software Applications

Fall 2021

Project Report – Lane detection

Made By

Andrew Yasser Shaker – 17P6069

Karim Mikhael Farid – 17P3061

Meriam Shoukry Fouad – 19P5870

Environment and tools used:

We worked on a python notebook using googlecolab as it provides GPU processing useful for training the model faster than normal CPUs and allows easy collaborative sharing of progress. We mounted google drive account that has the data used (training data, labels and 2 test videos)

Packages:

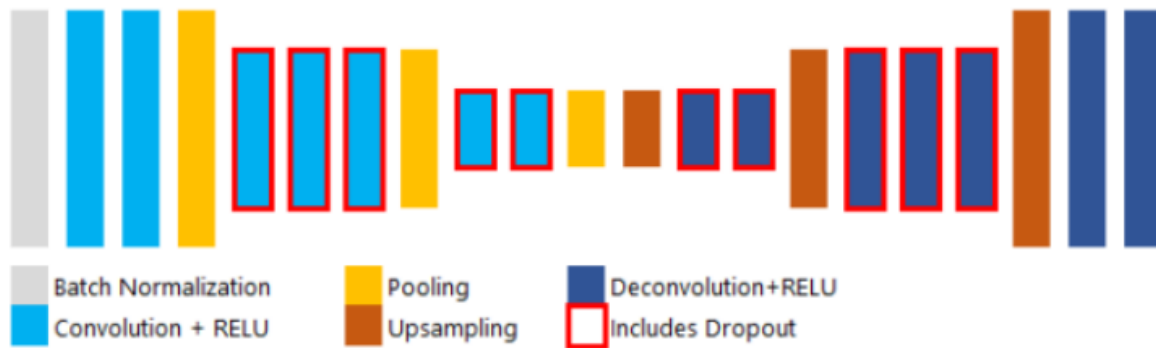
First, We started by importing the libraries and packages we will be using throughout the project from Keras (e.g.helper functions used to build the model like Conv2D, MaxPooling, Conv2DTranspose).

Model creation:

We then started creating our model sequentially by adding up the layers of architecture we used. The model consists of 21 layers.

- Layer 1 consists of a batch normalization
- Layers 2 and 3 perform adding convolution with a ReLu function.
- Layer 4 is a pooling layer.
- Layers 5, 6, and 7 perform adding convolution with a ReLu function and also includes dropout.
- Layer 8 is a pooling layer.
- Layers 9 and 10 perform adding convolution with a ReLu function and also includes dropout.
- Layer 11 is a pooling layer.
- Layer 12 is an Upsampling layer.
- Layers 13 and 14 perform adding deconvolution with a ReLu function and also includes dropout.
- Layer 15 is an Upsampling layer.
- Layers 16, 17, and 18 perform adding deconvolution with a ReLu function and also includes dropout.
- Layer 19 is an Upsampling layer.
- Layers 20 and 21 perform adding deconvolution with a ReLu function.

the architecture of the model is shown here:



We start by doing Batch normalization which is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This results in stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

convolutional layers were used with valid padding, strides of 1 and RELU activation function. The RELU activation function stands for rectified linear unit is the most popular activation function right now that makes deep NNs train faster now.

2 x 2 max pooling (down sampling) was used, as most researchers use max pooling instead of average pooling because it's been found in a lot of experiments to work well. max pooling has a set of hyperparameters but it has no parameters to learn. We set the size to 2 x 2 and leaving the default valid with no strides.

Dropout is a regularization method that is implemented per some layers in our model. It features a new hyperparameter for us that specifies the probability at which outputs of the layer are dropped out. We set that probability to 0.2 after researching.

We used 2 x2 upsampling by UpSampling2D function for the upsampling layer which is just a simple scaling up of the image by using nearest neighbour(default). We used Conv2DTranspose and this is a convolution operation which has a kernel that is learnt (just like normal conv2d operation) while training the model. Using Conv2DTranspose will also upsample its input

but the key difference is the model should learn what is the best upsampling for the job.

This architecture belongs is similar to a Unet, but without "skip" connections. It has a Contracting path (Encoder containing downsampling steps) and Expanding path (Decoder containing upsampling steps).

The architecture of the model is in a function called `create_model` that takes input shape and pool size as parameter and returns the model.

Loading the data:

We then loaded the training images from the google drive by `pickle.load()` and entered its path, we did the same for the training labels, then we transformed each to a numpy array to deal with easily.

For knowing the dimensions of training data we used `shape` method. It showed (12764, 80, 160, 3) meaning that there are 12764 images each measures 80 x 160 and 3 color channels (RGB).

Create the train and test splits:

We shuffled the images with their labels, and splited the data into training/validation using 10% for validation.

Proprocceasing:

We made some data augmentation using the `ImageDataGenerator`. Data augmentation is used to create new data from available training data artificially by adding new copies of available data but with certain changes. We tried horizontal flips, rescaling, rotations, shifts along vertical and horizontal axes all with multiple values and we also tried feature wise centering, but all of that was not useful to the model. So, we just used channel shifts with value of 20 and a brightness range for [0.2,1] to try to make the model robust in shaded areas.

Training the model and hyperparameters tuning:

We trained the models with various combinations of hyperparameters. We tried different batch sizes (140,130,120,110,100,90,80,70,60) and found that batch size of 70 is the optimum. We also tried two loss functions: binary cross entropy and mean square error and found that the mean square error performed better. We used Adam optimizer and we trained the model using 6,10, and 20 epochs but found that both 10 and 20 epochs nearly have the same result.

Model parameters:

We used model.summary to see the total parameters and the learnable parameters at each layer. Then saved the model to be tested on the test videos using the test script

```
=====
Total params: 181,693
Trainable params: 181,687
Non-trainable params: 6
=====
```

```
# Save model architecture and weights
model.save('full_CNN_model.h5')

# Show summary of model
model.summary()
```

Model: "sequential"

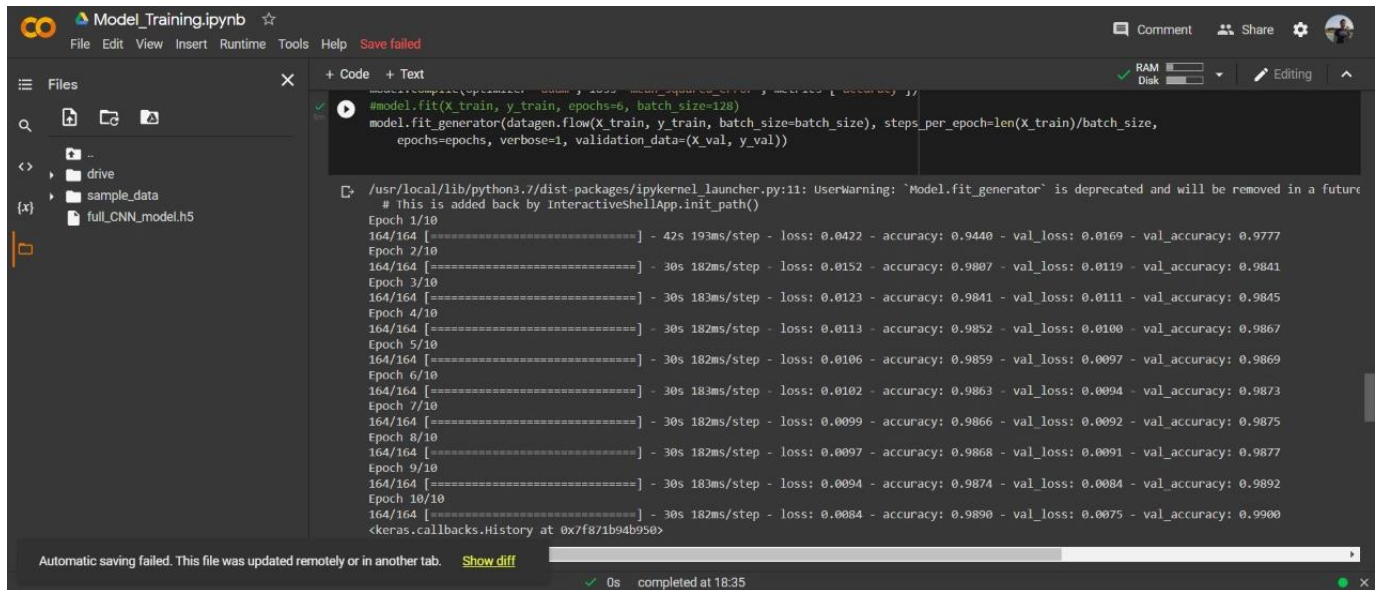
| Layer (type) | Output Shape | Param # |
|---|---------------------|---------|
| ===== | | |
| batch normalization (Batch Normalization) | (None, 80, 160, 3) | 12 |
| conv2d (Conv2D) | (None, 78, 158, 8) | 224 |
| conv2d_1 (Conv2D) | (None, 76, 156, 16) | 1168 |
| max_pooling2d (MaxPooling2D) | (None, 38, 78, 16) | 0 |
| conv2d_2 (Conv2D) | (None, 36, 76, 16) | 2320 |
| dropout (Dropout) | (None, 36, 76, 16) | 0 |
| conv2d_3 (Conv2D) | (None, 34, 74, 32) | 4640 |

| | | |
|--------------------------------|--------------------|-------|
| conv2d_3 (Conv2D) | (None, 34, 74, 32) | 4640 |
| dropout_1 (Dropout) | (None, 34, 74, 32) | 0 |
| conv2d_4 (Conv2D) | (None, 32, 72, 32) | 9248 |
| dropout_2 (Dropout) | (None, 32, 72, 32) | 0 |
| max_pooling2d_1 (MaxPooling2D) | (None, 16, 36, 32) | 0 |
| conv2d_5 (Conv2D) | (None, 14, 34, 64) | 18496 |
| dropout_3 (Dropout) | (None, 14, 34, 64) | 0 |
| conv2d_6 (Conv2D) | (None, 12, 32, 64) | 36928 |
| dropout_4 (Dropout) | (None, 12, 32, 64) | 0 |
| max_pooling2d_2 (MaxPooling2D) | (None, 6, 16, 64) | 0 |

| | | |
|--------------------------------------|--------------------|-------|
| max_pooling2d_2 (MaxPooling2D) | (None, 6, 16, 64) | 0 |
| up_sampling2d (UpSampling2D) | (None, 12, 32, 64) | 0 |
| conv2d_transpose (Conv2DTranspose) | (None, 14, 34, 64) | 36928 |
| dropout_5 (Dropout) | (None, 14, 34, 64) | 0 |
| conv2d_transpose_1 (Conv2DTranspose) | (None, 16, 36, 64) | 36928 |
| dropout_6 (Dropout) | (None, 16, 36, 64) | 0 |
| up_sampling2d_1 (UpSampling2D) | (None, 32, 72, 64) | 0 |
| conv2d_transpose_2 (Conv2DTranspose) | (None, 34, 74, 32) | 18464 |
| dropout_7 (Dropout) | (None, 34, 74, 32) | 0 |
| conv2d_transpose_3 (Conv2DTranspose) | (None, 36, 76, 32) | 9248 |
| dropout_8 (Dropout) | (None, 36, 76, 32) | 0 |
| conv2d_transpose_4 (Conv2DTranspose) | (None, 38, 78, 16) | 4624 |

| | | |
|--------------------------------------|---------------------|------|
| dropout_7 (Dropout) | (None, 34, 74, 32) | 0 |
| conv2d_transpose_3 (Conv2DTranspose) | (None, 36, 76, 32) | 9248 |
| dropout_8 (Dropout) | (None, 36, 76, 32) | 0 |
| conv2d_transpose_4 (Conv2DTranspose) | (None, 38, 78, 16) | 4624 |
| dropout_9 (Dropout) | (None, 38, 78, 16) | 0 |
| up_sampling2d_2 (UpSampling2D) | (None, 76, 156, 16) | 0 |
| conv2d_transpose_5 (Conv2DTranspose) | (None, 78, 158, 16) | 2320 |
| conv2d_transpose_6 (Conv2DTranspose) | (None, 80, 160, 1) | 145 |
| ===== | | |
| Total params: 181,693 | | |
| Trainable params: 181,687 | | |
| Non-trainable params: 6 | | |

Screenshot from model training:



```
Model_Training.ipynb
File Edit View Insert Runtime Tools Help Save failed

+ Code + Text
#model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
#model.fit(X_train, y_train, epochs=6, batch_size=128)
model.fit_generator(datagen.flow(X_train, y_train, batch_size=batch_size), steps_per_epoch=len(X_train)/batch_size,
                    epochs=epochs, verbose=1, validation_data=(X_val, y_val))

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:11: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future
# This is added back by InteractiveShellApp.init_path()
Epoch 1/10
164/164 [=====] - 42s 193ms/step - loss: 0.0422 - accuracy: 0.9440 - val_loss: 0.0169 - val_accuracy: 0.9777
Epoch 2/10
164/164 [=====] - 30s 182ms/step - loss: 0.0152 - accuracy: 0.9807 - val_loss: 0.0119 - val_accuracy: 0.9841
Epoch 3/10
164/164 [=====] - 30s 183ms/step - loss: 0.0123 - accuracy: 0.9841 - val_loss: 0.0111 - val_accuracy: 0.9845
Epoch 4/10
164/164 [=====] - 30s 182ms/step - loss: 0.0113 - accuracy: 0.9852 - val_loss: 0.0100 - val_accuracy: 0.9867
Epoch 5/10
164/164 [=====] - 30s 182ms/step - loss: 0.0106 - accuracy: 0.9859 - val_loss: 0.0097 - val_accuracy: 0.9869
Epoch 6/10
164/164 [=====] - 30s 183ms/step - loss: 0.0102 - accuracy: 0.9863 - val_loss: 0.0094 - val_accuracy: 0.9873
Epoch 7/10
164/164 [=====] - 30s 182ms/step - loss: 0.0099 - accuracy: 0.9866 - val_loss: 0.0092 - val_accuracy: 0.9875
Epoch 8/10
164/164 [=====] - 30s 182ms/step - loss: 0.0097 - accuracy: 0.9868 - val_loss: 0.0091 - val_accuracy: 0.9877
Epoch 9/10
164/164 [=====] - 30s 183ms/step - loss: 0.0094 - accuracy: 0.9874 - val_loss: 0.0084 - val_accuracy: 0.9892
Epoch 10/10
164/164 [=====] - 30s 182ms/step - loss: 0.0084 - accuracy: 0.9890 - val_loss: 0.0075 - val_accuracy: 0.9900
<keras.callbacks.History at 0x7f871b94b950>

Automatic saving failed. This file was updated remotely or in another tab. Show diff
0s completed at 18:35
```

Links to test videos output: (please copy and paste in browser if not working)

<https://drive.google.com/drive/folders/18qg7PHCyjyvDLIb7sqhK2PA4FaaMR8NW?usp=sharing>