Netflix Data Science Boot Camp

Detecting Kid Friendly Videos

By Alex, Vu, and Kwabena

Can we detect if a video is kid friendly

Youtube surprisingly doesn't have a known bot to detect videos for kids.

The Approach

Initial approach

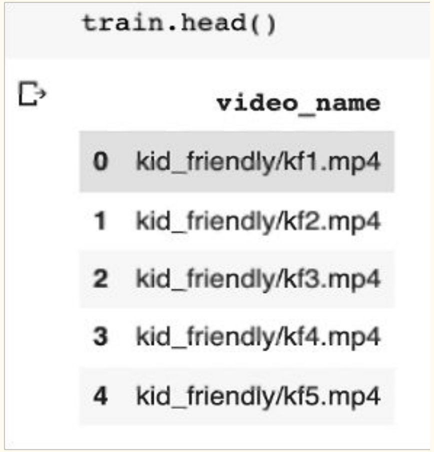
- Based on a tutorial to build our own Video Classification Model
- Utilized the UCF101 action training set.
- Over a 100 different actions ranging anywhere from applying lipstick to yo-yo-ing
- Retrofitted and applied to our problem.

DATASET

Collection of Data

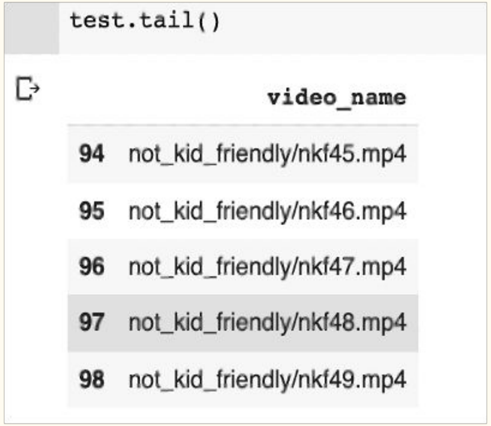
- ✓ Downloading videos from YouTube → 50 Kid Friendly, 50 Not Kid Friendly

- ✓ Storage of data (videos) on a database
- ✓ Creation of train and test .txt files
- ✓ Importation of training and testing videos .txt files into Google Colab



```
train.head()

        video_name
0  kid_friendly/kf1.mp4
1  kid_friendly/kf2.mp4
2  kid_friendly/kf3.mp4
3  kid_friendly/kf4.mp4
4  kid_friendly/kf5.mp4
```

Train List DataFrame



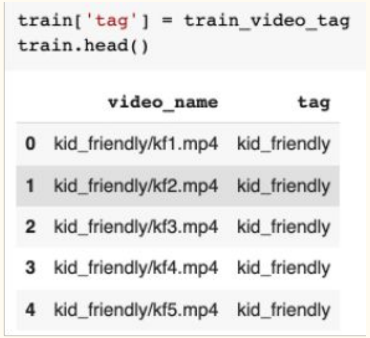
```
test.tail()

                video_name
94  not_kid_friendly/nkf45.mp4
95  not_kid_friendly/nkf46.mp4
96  not_kid_friendly/nkf47.mp4
97  not_kid_friendly/nkf48.mp4
98  not_kid_friendly/nkf49.mp4
```

Test List DataFrame

DATASET CONT'D

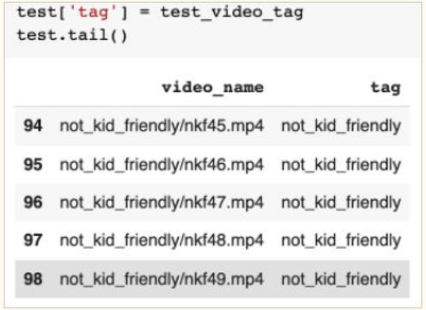
Processing of Data

1. Addition of tags



```
train['tag'] = train_video_tag
train.head()
```

	video_name	tag
0	kid_friendly/kf1.mp4	kid_friendly
1	kid_friendly/kf2.mp4	kid_friendly
2	kid_friendly/kf3.mp4	kid_friendly
3	kid_friendly/kf4.mp4	kid_friendly
4	kid_friendly/kf5.mp4	kid_friendly

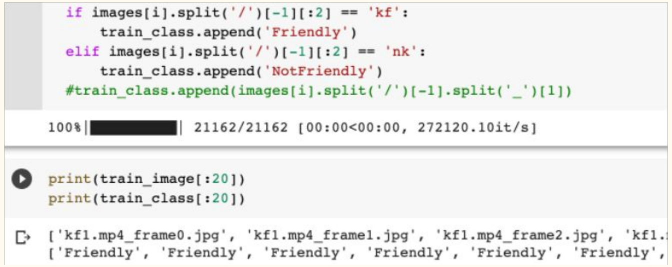


```
test['tag'] = test_video_tag
test.tail()
```

	video_name	tag
94	not_kid_friendly/nkf45.mp4	not_kid_friendly
95	not_kid_friendly/nkf46.mp4	not_kid_friendly
96	not_kid_friendly/nkf47.mp4	not_kid_friendly
97	not_kid_friendly/nkf48.mp4	not_kid_friendly
98	not_kid_friendly/nkf49.mp4	not_kid_friendly

2. Extraction and storage of frames from training videos

3. Creation of Class for each video frame

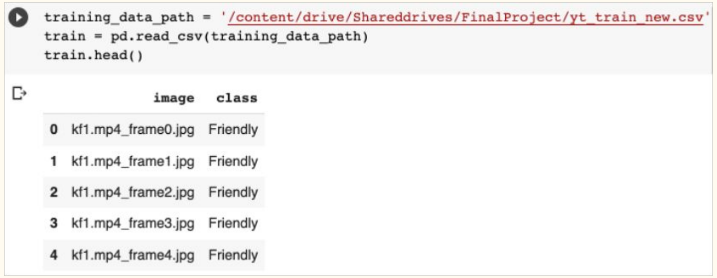


```
    if images[i].split('/')[-1][:2] == 'kf':
        train_class.append('Friendly')
    elif images[i].split('/')[-1][:2] == 'nk':
        train_class.append('NotFriendly')
    #train_class.append(images[i].split('/')[-1].split('_')[1])
```

```
100%|██████████| 21162/21162 [00:00<00:00, 272120.10it/s]
```

```
print(train_image[:20])
print(train_class[:20])
```

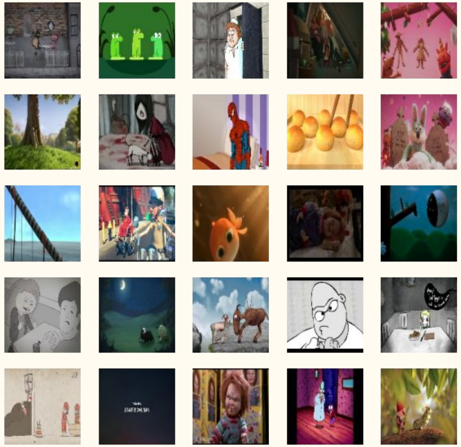
```
['kf1.mp4_frame0.jpg', 'kf1.mp4_frame1.jpg', 'kf1.mp4_frame2.jpg', 'kf1.
['Friendly', 'Friendly', 'Friendly', 'Friendly', 'Friendly', 'Friendly',
```

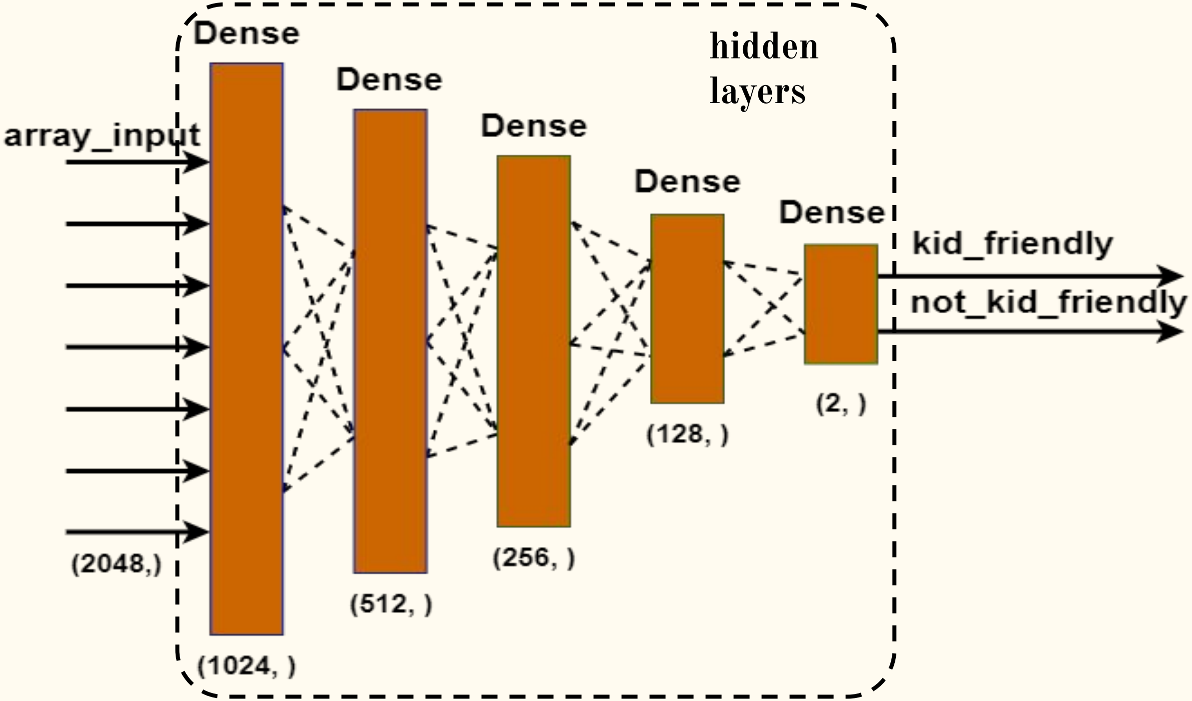


```
training_data_path = '/content/drive/Shareddrives/FinalProject/yt_train_new.csv'
train = pd.read_csv(training_data_path)
train.head()
```

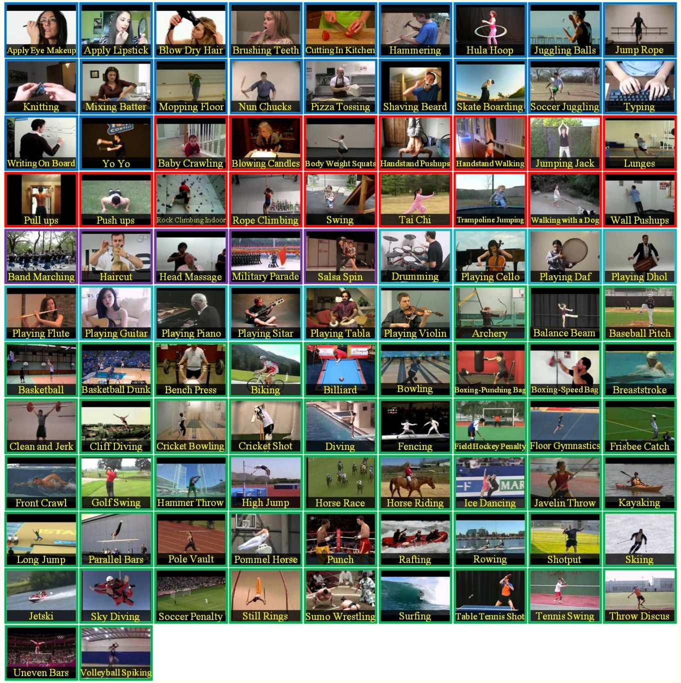
	image	class
0	kf1.mp4_frame0.jpg	Friendly
1	kf1.mp4_frame1.jpg	Friendly
2	kf1.mp4_frame2.jpg	Friendly
3	kf1.mp4_frame3.jpg	Friendly
4	kf1.mp4_frame4.jpg	Friendly

Deep learning model - Fully connected neural network

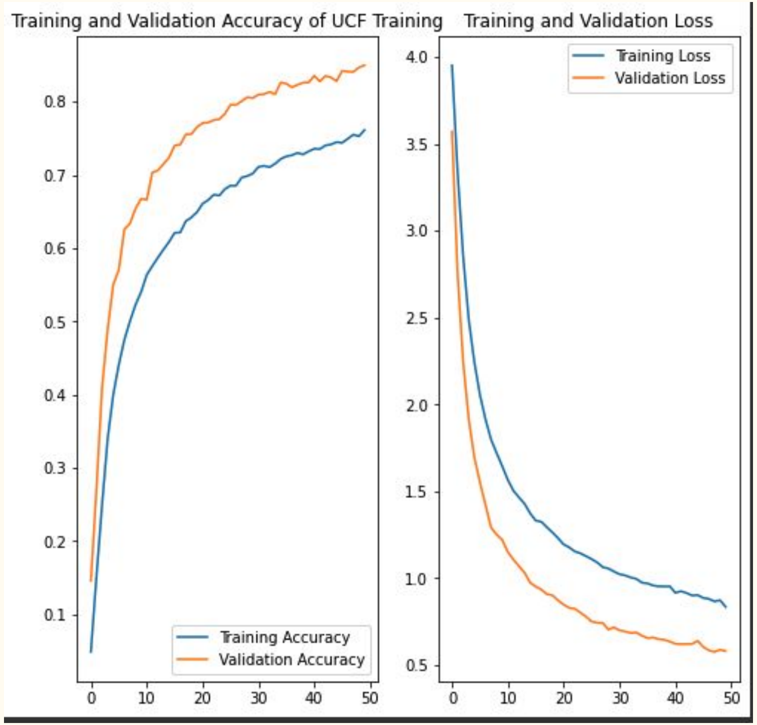




Deep learning model

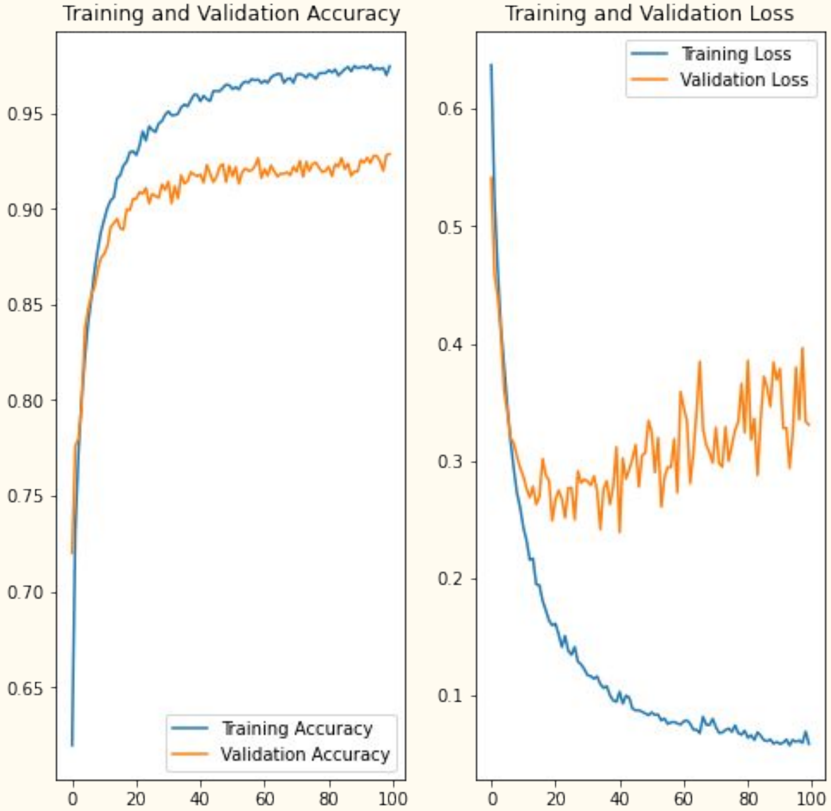


UCF101 - Action Recognition Data Set



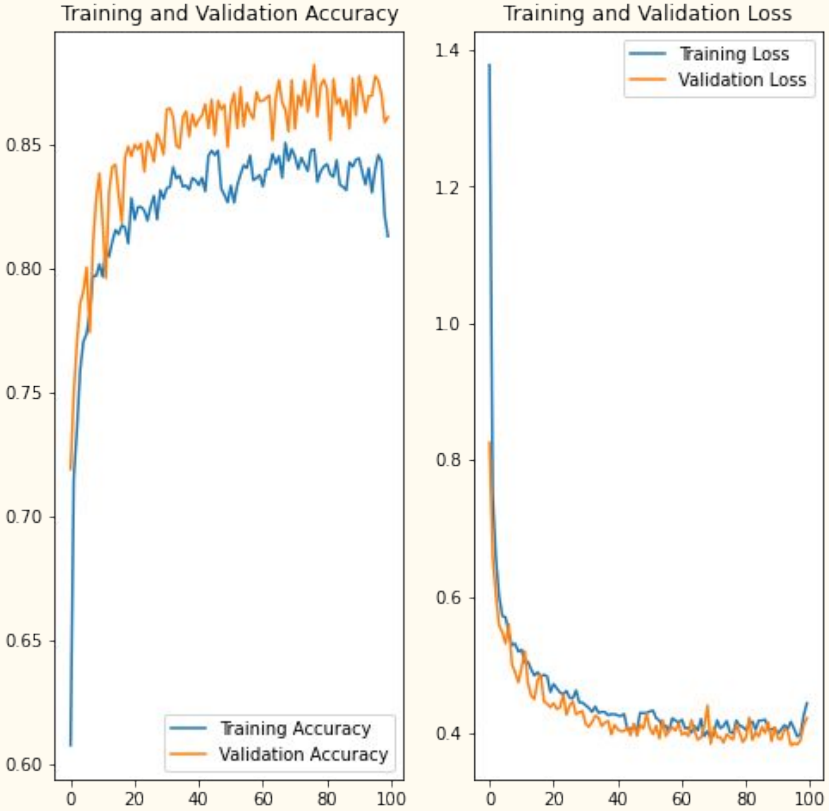
Training and Validation Result from [3]

Deep learning model with dense hidden layers



Training and Validation Accuracy result

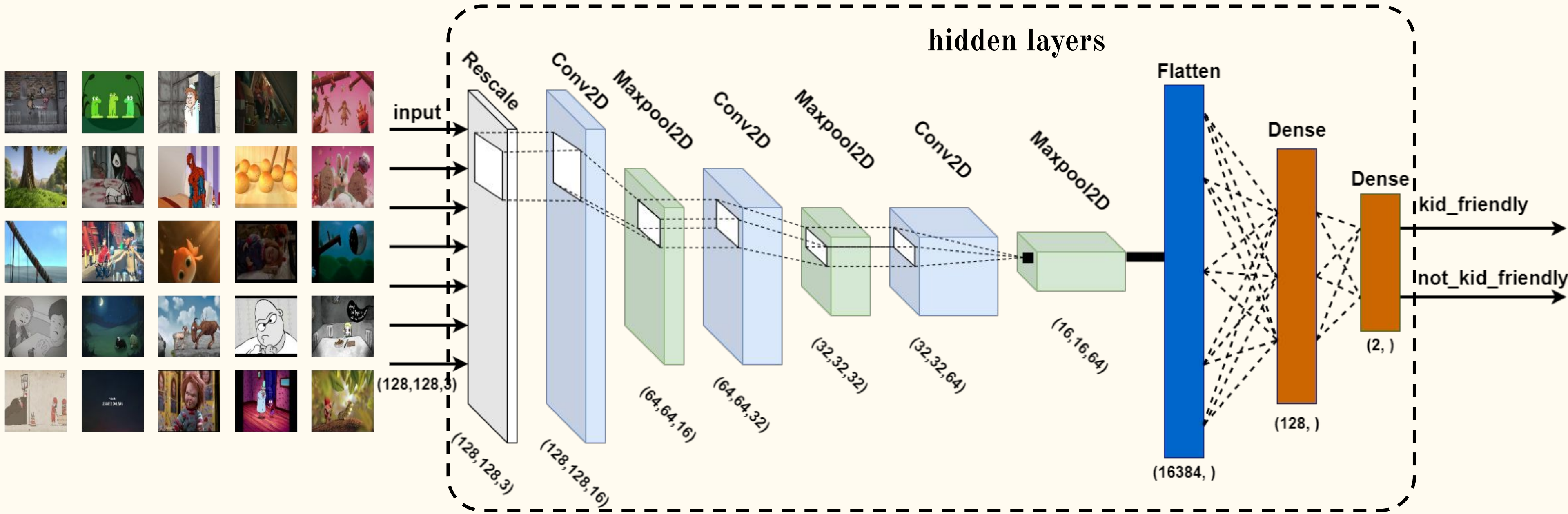
without regularizers



Training and Validation Accuracy result

with L2 regularizers

Deep learning model - Convolution neural network

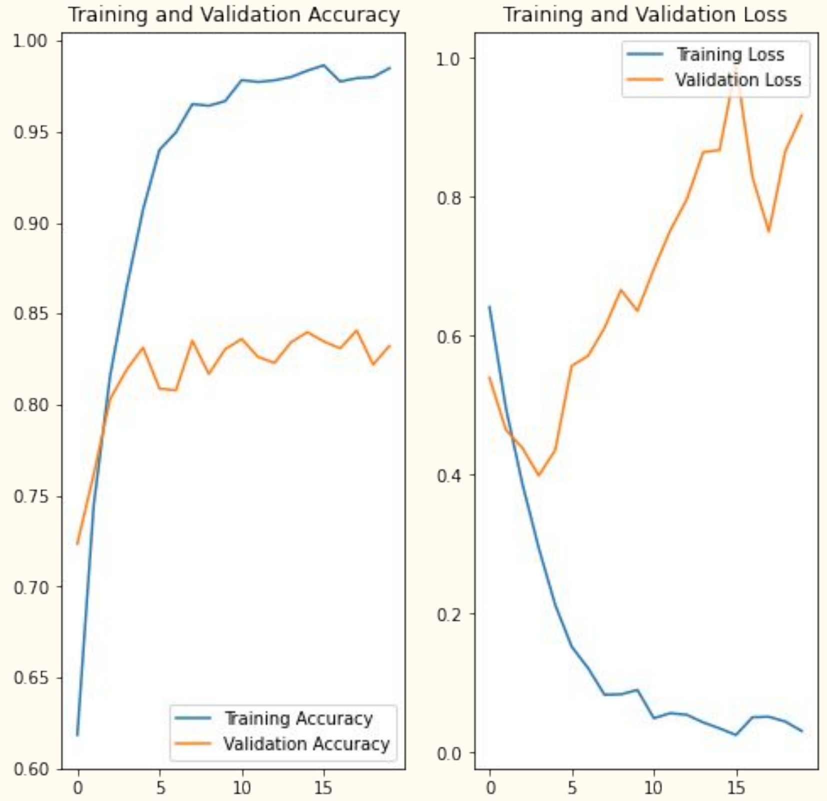


Total parameters: 2,121,122

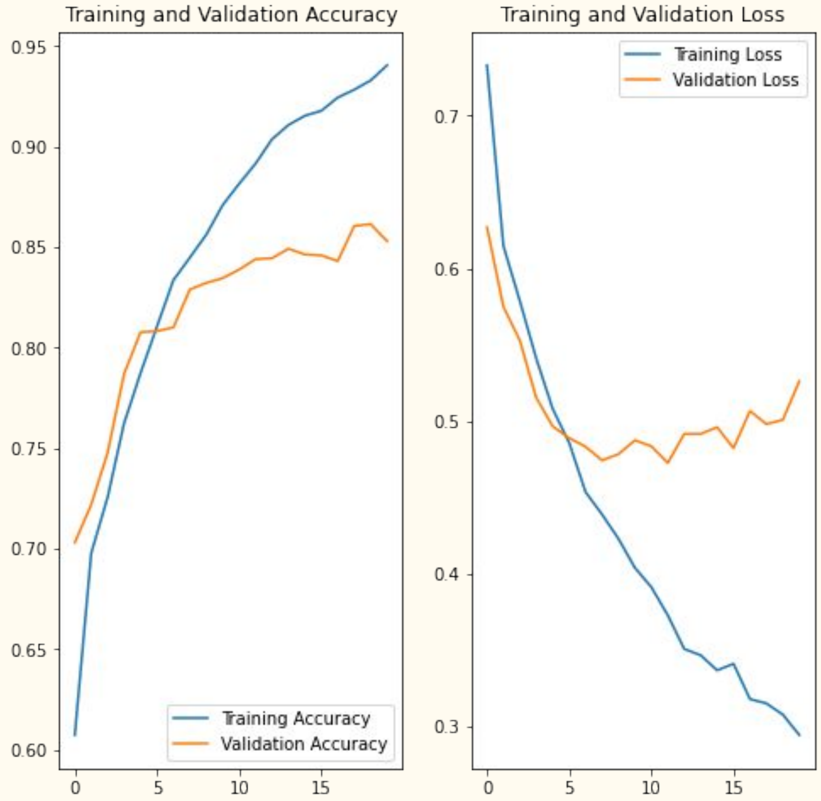
Deep learning model - Convolution neural network

Layer	**Output shape**	**Parameters**
Rescaling	(128, 128, 3)	0
Conv2D	(128, 128, 16)	448
Maxpool2D	(64, 64, 16)	0
Conv2D	(64, 64, 32)	4,640
Maxpool2D	(32, 32, 32)	0
Conv2D	(32, 32, 64)	18,496
Maxpool2D	(16, 16, 64)	0
Flatten	(, 16384)	0
Dense	(, 128)	2,097,280
Dense	(, 2)	258
	Total	**2,121,122**

Deep learning model with CNN layers



Training and Validation Accuracy result



Training and Validation Accuracy result

with Dropout layers and L2 regularizers

Conclusion & Discussion

- Dense Layers deep neural network model yields accuracy of 85% for split of 80-20 training-validating dataset.
- CNN deep neural network experiences overfitting which can be improved by adding Dropout layers and Regularizers.
- Noise comes from labelling the dataset within videos → Use sub-model to support labelling process

Thank you!

References

[1] Netflix Data Science Boot camp materials

[2] TensorFlow, Image Classification Tutorial, https://www.tensorflow.org/tutorials/images/classification

[3] Step-by-Step Deep Learning Tutorial to Build your own Video Classification Model, https://www.analyticsvidhya.com/blog/2019/09/step-by-step-deep-learning-tutorial-video-classification-python/