CNN Model to Detect Fire in Images

BITS F312 Neural Networks and Fuzzy Logic - Assignment 1

Team 9
Aryan Gupta - 2019A7PS0017G
Tarush Agarwal - 2019A7PS0025G
Tanmay Patil - 2019A7PS0054G

Data setup

- Took a sample of about 12k images (one in every 6th frame)
- Image size: 128x128 (pixels) - stored in numpy arrays
- No Augmentation

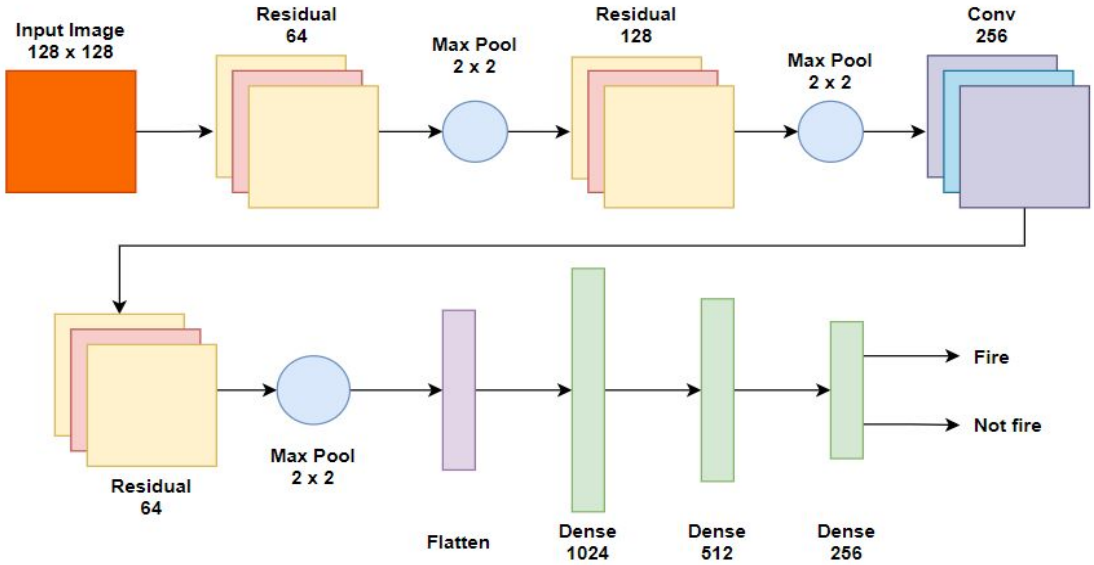
Train-Valid Split of input data:

- Out-of-Sample Data split
- num_train ≈ 75%, num_valid ≈ 25%
    - 19 : 1 in first 4k [num_train += 3800, num_val += 200]
    - 11 : 1 in next 4k [num_train += 2750, num_val += 250]
    - 1 : 19 in next 1k [num_train += 50, num_val += 950]
    - 1 : 2 in last 3k [num_train += 1000, num_val += 2000]

Model Architecture

Basic idea inspired from VGG16 and ResNet models:

- Convolutional layers alternated with Max Pool layers
- Used Residual Modules in Convolutional Layers
- Followed by Flattening of the Feature Maps
- “Dense” Regularised Dense Layers alternated with Dropout layers



Model Architecture - Convolutions

- Convolutional layers - neither too shallow, nor too deep
- Dense Residual modules with two Convolutional Layers each
- Residual modules alternated with MaxPool Layers (2x2)
- Additional Convolutional Layer
- Activation Functions used:
  ReLU in Convolutional Layers, ReLU and Linear in Residual Module
- Did not use Inception modules

```
# conv layers
convOut = residual_module(inp, 64)
convOut = MaxPool2D((2, 2))(convOut)
convOut = residual_module(convOut, 128)
convOut = MaxPool2D((2, 2))(convOut)
convOut = Conv2D(256, (3, 3), padding='valid', activation='relu', input_shape=target_size)(convOut)
convOut = residual_module(convOut, 64)
convOut = MaxPool2D((2, 2))(convOut)
```

Model Architecture - Dense Layers

- Not-so-deep Dense Layers
- L2 Regularisation with Lambda = 0.01
- Dropout layers after each dense layer
- Activation Functions used:
  ReLU in hidden layers and Sigmoid in Final layer
- Loss function used: Binary cross Entropy

```
# dense layers
denseOut = Dense(1024, activation=tf.nn.relu, kernel_regularizer=tf.keras.regularizers.L2(0.01))(flatOut)
denseOut = Dropout(0.2)(denseOut)
denseOut = Dense(512, activation=tf.nn.relu, kernel_regularizer=tf.keras.regularizers.L2(0.01))(denseOut)
denseOut = Dropout(0.2)(denseOut)
denseOut = Dense(64, activation=tf.nn.relu, kernel_regularizer=tf.keras.regularizers.L2(0.01))(denseOut)
denseOut = Dropout(0.1)(denseOut)
out = Dense(1, activation="sigmoid")(denseOut)
```

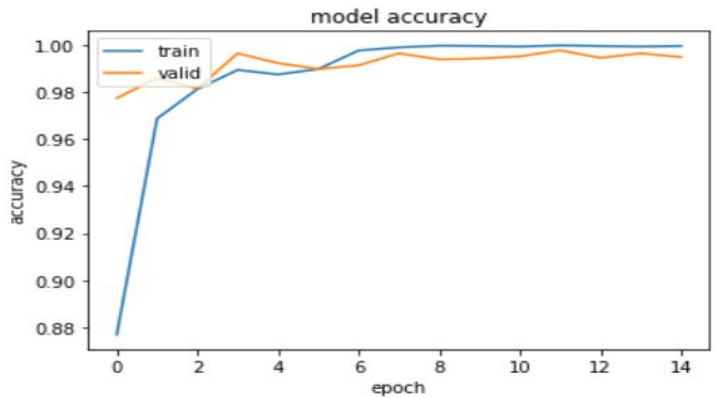
Training of the Model

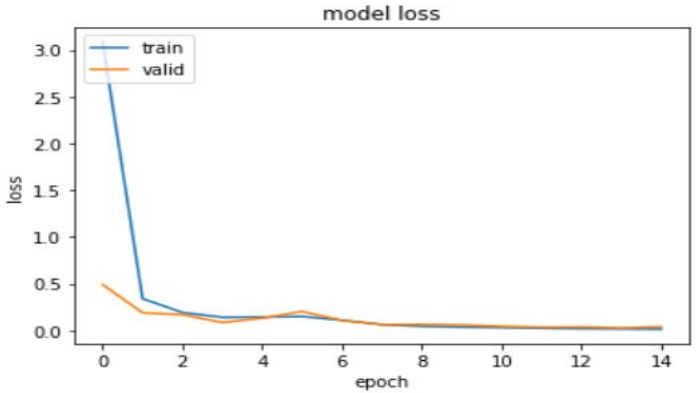
- Optimizer: Adam with initial learning rate 0.001
- Batch-processing with batch-size: 16
- No. of epochs: 15 to prevent overfitting
- Callback: Reduce Learning rate on plateau

```
history = model.fit(x_train, y_train, validation_data=(x_valid, y_valid), epochs=15, batch_size=16, callbacks = callbacks)
```

Results

Training the model - the accuracy and loss curves are plotted:





- Training accuracy and Validation accuracies: 99+ %
- Score on the Public and Private leaderboards: 90+ %

Things that didn’t work out..

Expected Failures:

- 192x192 performed worse
- Decreasing number of features
- Very deep convolutional layers degrades performance

Unexpected Failures:

- Inception module not helping
- Early stopping didn’t help

Thank You!

Link to the NB: https://www.kaggle.com/code/tanmayrajendrapatil/team9-nnfl-assgn1-new-loss-fn