Activity Recognition from Video

Data:

Videos taken for 7 different activities (3-Similar, 4-Different) from UCF-101 Dataset:

1. Apply Eye Makeup (Similar)
2. Apply Lipstick (Similar)
3. Brushing Teeth (Similar)
4. Basketball (Different)
5. Diving (Different)
6. Nunchucks (Different)
7. Punch (Different)

Train Data:

8 videos/activity ⇔ 500 frames/video => 4,000 frames/activity
Total frames in training dataset: ~ 28,000

Cross-Validation Data:

10 videos/activity ⇔ 100 frames/video => 1,000 frames/activity
Total frames in CV dataset: ~ 7,000

Test-Data Data:

10 videos/activity ⇔ 100 frames/video => 1,000 frames/activity
Total frames in CV dataset: ~ 7,000
Video resized to size: (224, 224, 3) (RGB images)

Classification	Sequential
Train Shape:	Train Shape:
(29566, 224, 224, 3)	(14783, 16, 224, 224, 3) x 2
(29566, 7)	(14783, 16, 7) x 2
CV Shape:	CV Shape:
(7367, 224, 224, 3)	(7351, 16, 224, 224, 3)
(7367, 7)	(7351, 16, 7)
Test Shape:	Test Shape:
(7320, 224, 224, 3)	(7304, 16, 224, 224, 3)
(7320, 7)	(7304, 16, 7)

Due to memory constraints on Henry Cluster, LSTM is trained in 2 batches each with ~14,000 frames.

Approach 1:

Network:

(https://github.com/LisaAnne/lisa-caffe-public/blob/lstm_video_deploy/examples/LRCN_activity_recognition/train_test_lstm_RGB.prototxt)

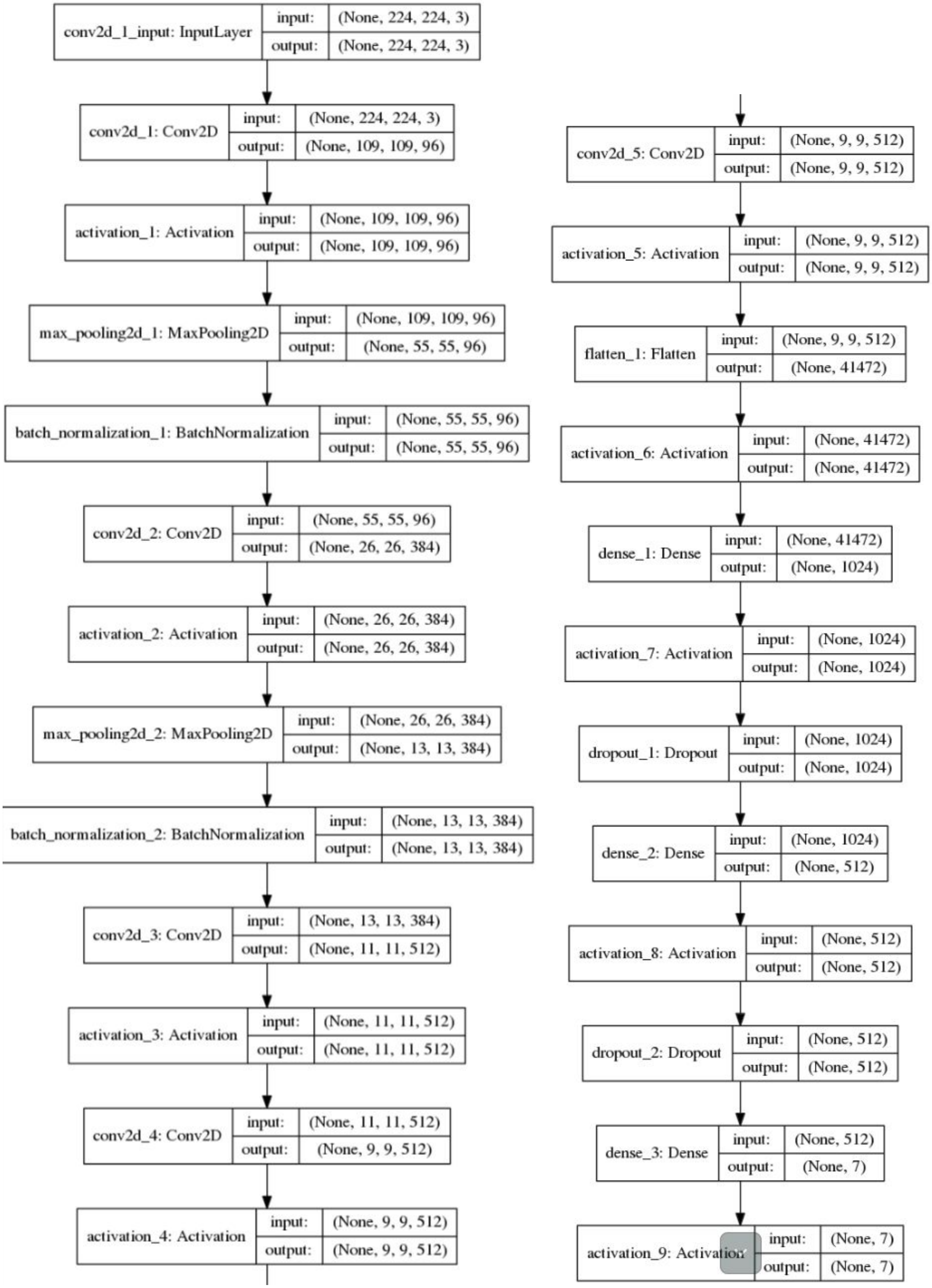
Classification Model:

- 5 x Convolutional Layers
- 2 x Max Pooling Layers
- 2 x Batch Normalization
- 2 x Fully connected layers (1024 & 512)
- Activation: ReLU
- Classification: Softmax
- Loss: Categorical Cross Entropy
- Optimizer: Adam

- Total parameters: 48,327,303
- Total Trainable parameters: 48,326,343

Hyper-parameters:

- Regularization: Dropout (probability: 0.5)
- Batch size: 128
- Learning Rate: 1e-4
- Decay Rate: 1e-2


conv2d_1_input: InputLayer | input: (None, 224, 224, 3) | output: (None, 224, 224, 3)
conv2d_1: Conv2D | input: (None, 224, 224, 3) | output: (None, 109, 109, 96)
activation_1: Activation | input: (None, 109, 109, 96) | output: (None, 109, 109, 96)
max_pooling2d_1: MaxPooling2D | input: (None, 109, 109, 96) | output: (None, 55, 55, 96)
batch_normalization_1: BatchNormalization | input: (None, 55, 55, 96) | output: (None, 55, 55, 96)
conv2d_2: Conv2D | input: (None, 55, 55, 96) | output: (None, 26, 26, 384)
activation_2: Activation | input: (None, 26, 26, 384) | output: (None, 26, 26, 384)
max_pooling2d_2: MaxPooling2D | input: (None, 26, 26, 384) | output: (None, 13, 13, 384)
batch_normalization_2: BatchNormalization | input: (None, 13, 13, 384) | output: (None, 13, 13, 384)
conv2d_3: Conv2D | input: (None, 13, 13, 384) | output: (None, 11, 11, 512)
activation_3: Activation | input: (None, 11, 11, 512) | output: (None, 11, 11, 512)
conv2d_4: Conv2D | input: (None, 11, 11, 512) | output: (None, 9, 9, 512)
activation_4: Activation | input: (None, 9, 9, 512) | output: (None, 9, 9, 512)
conv2d_5: Conv2D | input: (None, 9, 9, 512) | output: (None, 9, 9, 512)
activation_5: Activation | input: (None, 9, 9, 512) | output: (None, 9, 9, 512)
flatten_1: Flatten | input: (None, 9, 9, 512) | output: (None, 41472)
activation_6: Activation | input: (None, 41472) | output: (None, 41472)
dense_1: Dense | input: (None, 41472) | output: (None, 1024)
activation_7: Activation | input: (None, 1024) | output: (None, 1024)
dropout_1: Dropout | input: (None, 1024) | output: (None, 1024)
dense_2: Dense | input: (None, 1024) | output: (None, 512)
activation_8: Activation | input: (None, 512) | output: (None, 512)
dropout_2: Dropout | input: (None, 512) | output: (None, 512)
dense_3: Dense | input: (None, 512) | output: (None, 7)
activation_9: Activation | input: (None, 7) | output: (None, 7)

Sequential Model:

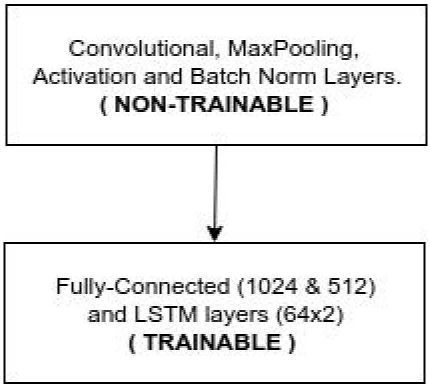
- 5 x Convolutional Layers
- 2 x Max Pooling Layers
- 2 x Batch Normalization
- 2 x Fully connected layers (1024 & 512)
- 2 x LSTM layers
- Activation: ReLU
- Classification: Softmax
- Loss: Categorical Cross Entropy
- Optimizer: Adam

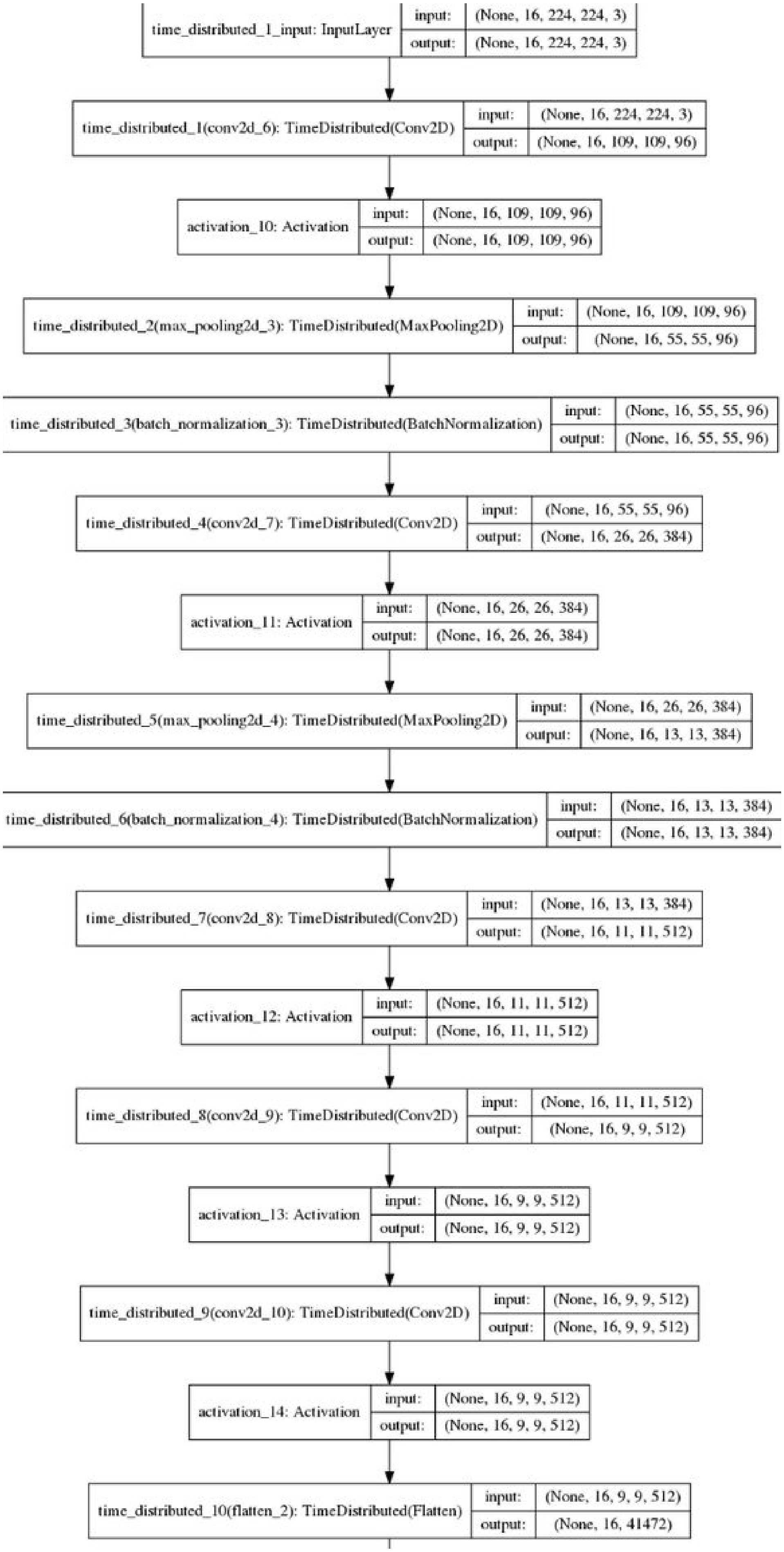
- Total parameters: 48,504,903
- Total Trainable parameters: 43,174,343

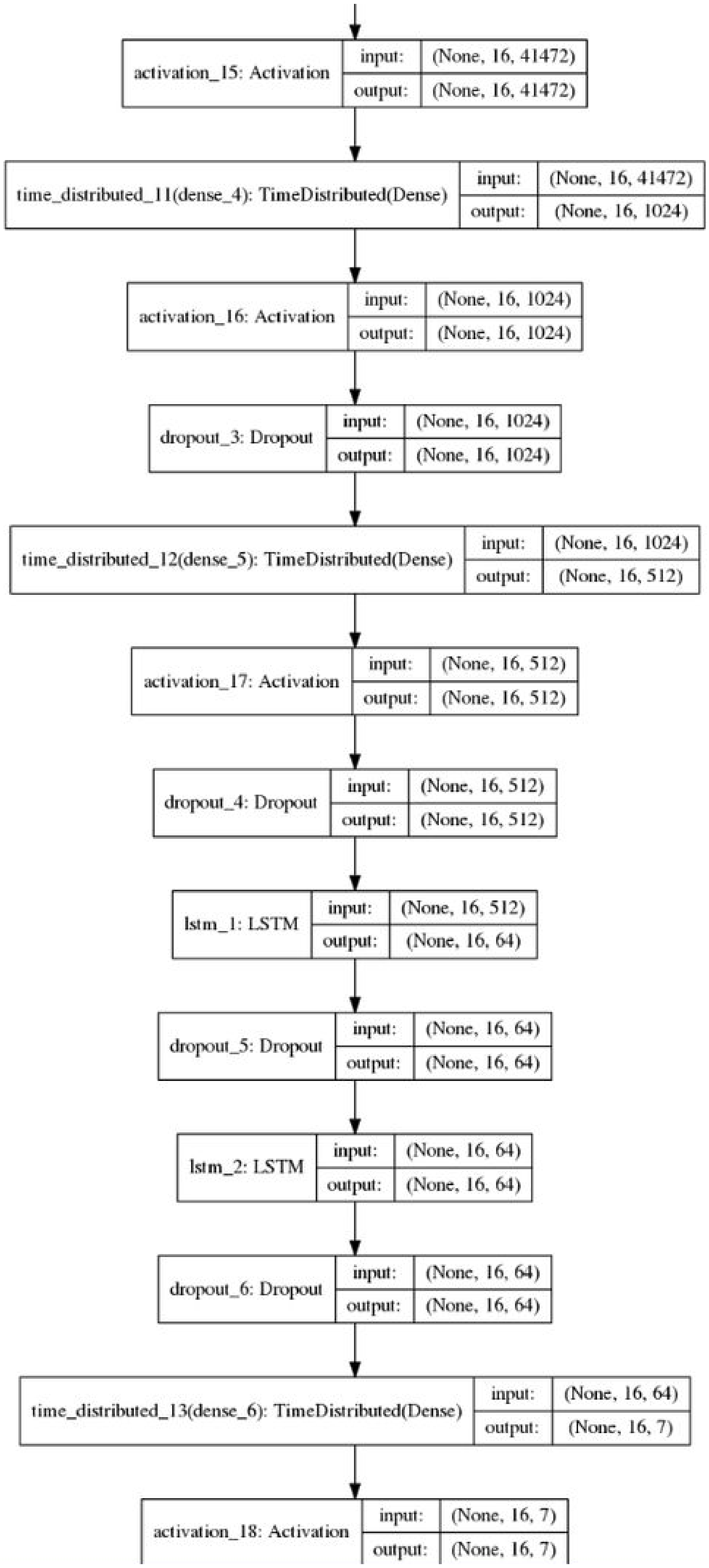
Hyper-parameters:

- Sequence Length: 16
- Regularization: Dropout (probability: 0.5)
- Batch size: 64
- Learning Rate: 1e-4
- Decay Rate: 1e-2
- LSTM cells: 64 cells x 2 layers

- Due to memory constraints on Henry cluster, batch size is reduced to 64.
- Since the model design is same as classification, we use the classification pretrained weights for initialization.
- Here the **convolutional part is frozen** whereas the **fully-connected layers and the LSTM layers are made trainable**. (i.e. until **Flatten** layer, all layers are **non-trainable**)


Convolutional, MaxPooling, Activation and Batch Norm Layers.
(NON-TRAINABLE)
Fully-Connected (1024 & 512) and LSTM layers (64x2)
(TRAINABLE)


time_distributed_1_input: InputLayer
input: (None, 16, 224, 224, 3)
output: (None, 16, 224, 224, 3)
time_distributed_1(conv2d_6): TimeDistributed(Conv2D)
input: (None, 16, 224, 224, 3)
output: (None, 16, 109, 109, 96)
activation_10: Activation
input: (None, 16, 109, 109, 96)
output: (None, 16, 109, 109, 96)
time_distributed_2(max_pooling2d_3): TimeDistributed(MaxPooling2D)
input: (None, 16, 109, 109, 96)
output: (None, 16, 55, 55, 96)
time_distributed_3(batch_normalization_3): TimeDistributed(BatchNormalization)
input: (None, 16, 55, 55, 96)
output: (None, 16, 55, 55, 96)
time_distributed_4(conv2d_7): TimeDistributed(Conv2D)
input: (None, 16, 55, 55, 96)
output: (None, 16, 26, 26, 384)
activation_11: Activation
input: (None, 16, 26, 26, 384)
output: (None, 16, 26, 26, 384)
time_distributed_5(max_pooling2d_4): TimeDistributed(MaxPooling2D)
input: (None, 16, 26, 26, 384)
output: (None, 16, 13, 13, 384)
time_distributed_6(batch_normalization_4): TimeDistributed(BatchNormalization)
input: (None, 16, 13, 13, 384)
output: (None, 16, 13, 13, 384)
time_distributed_7(conv2d_8): TimeDistributed(Conv2D)
input: (None, 16, 13, 13, 384)
output: (None, 16, 11, 11, 512)
activation_12: Activation
input: (None, 16, 11, 11, 512)
output: (None, 16, 11, 11, 512)
time_distributed_8(conv2d_9): TimeDistributed(Conv2D)
input: (None, 16, 11, 11, 512)
output: (None, 16, 9, 9, 512)
activation_13: Activation
input: (None, 16, 9, 9, 512)
output: (None, 16, 9, 9, 512)
time_distributed_9(conv2d_10): TimeDistributed(Conv2D)
input: (None, 16, 9, 9, 512)
output: (None, 16, 9, 9, 512)
activation_14: Activation
input: (None, 16, 9, 9, 512)
output: (None, 16, 9, 9, 512)
time_distributed_10(flatten_2): TimeDistributed(Flatten)
input: (None, 16, 9, 9, 512)
output: (None, 16, 41472)

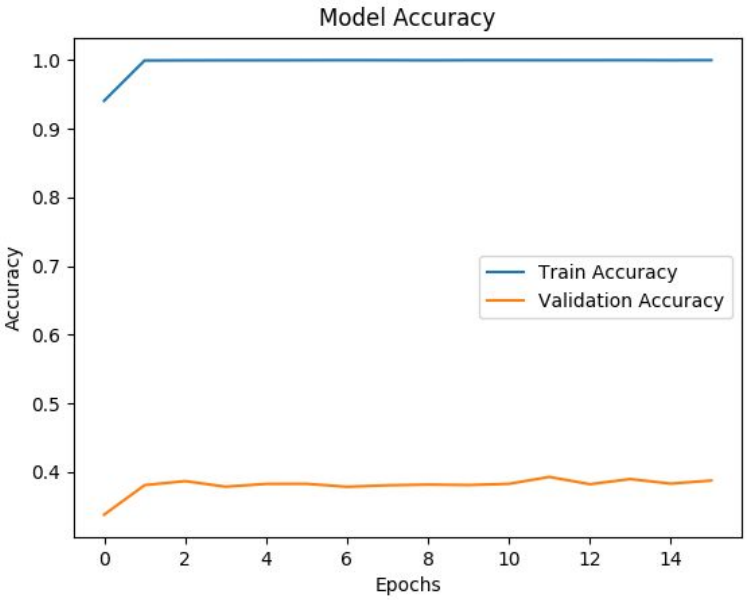

activation_15: Activation
input: (None, 16, 41472)
output: (None, 16, 41472)
time_distributed_11(dense_4): TimeDistributed(Dense)
input: (None, 16, 41472)
output: (None, 16, 1024)
activation_16: Activation
input: (None, 16, 1024)
output: (None, 16, 1024)
dropout_3: Dropout
input: (None, 16, 1024)
output: (None, 16, 1024)
time_distributed_12(dense_5): TimeDistributed(Dense)
input: (None, 16, 1024)
output: (None, 16, 512)
activation_17: Activation
input: (None, 16, 512)
output: (None, 16, 512)
dropout_4: Dropout
input: (None, 16, 512)
output: (None, 16, 512)
lstm_1: LSTM
input: (None, 16, 512)
output: (None, 16, 64)
dropout_5: Dropout
input: (None, 16, 64)
output: (None, 16, 64)
lstm_2: LSTM
input: (None, 16, 64)
output: (None, 16, 64)
dropout_6: Dropout
input: (None, 16, 64)
output: (None, 16, 64)
time_distributed_13(dense_6): TimeDistributed(Dense)
input: (None, 16, 64)
output: (None, 16, 7)
activation_18: Activation
input: (None, 16, 7)
output: (None, 16, 7)

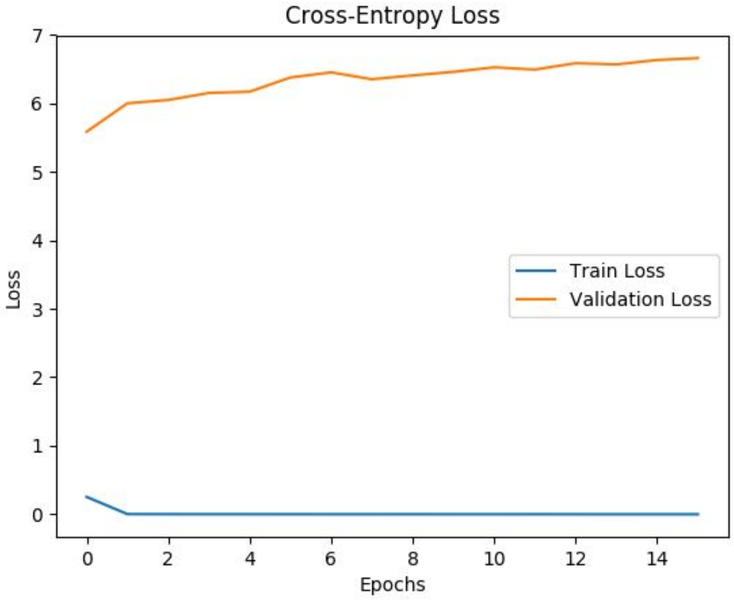
Result:

Classification:

Train Loss	:	0.00029122
Cross Validation Loss	:	6.6589
Test Loss	:	5.7841

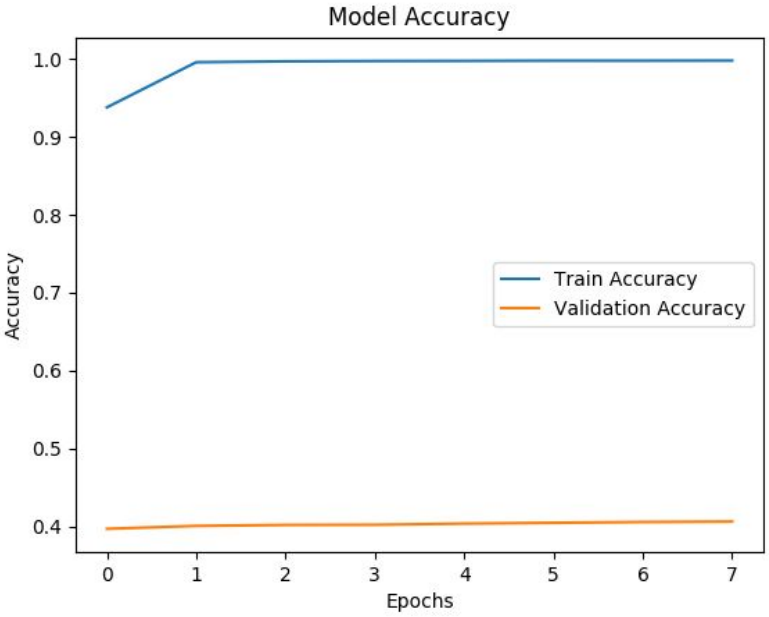
Train Accuracy	:	100.00 %
Cross Validation Accuracy	:	38.75 %
Test Accuracy	:	46.95 %

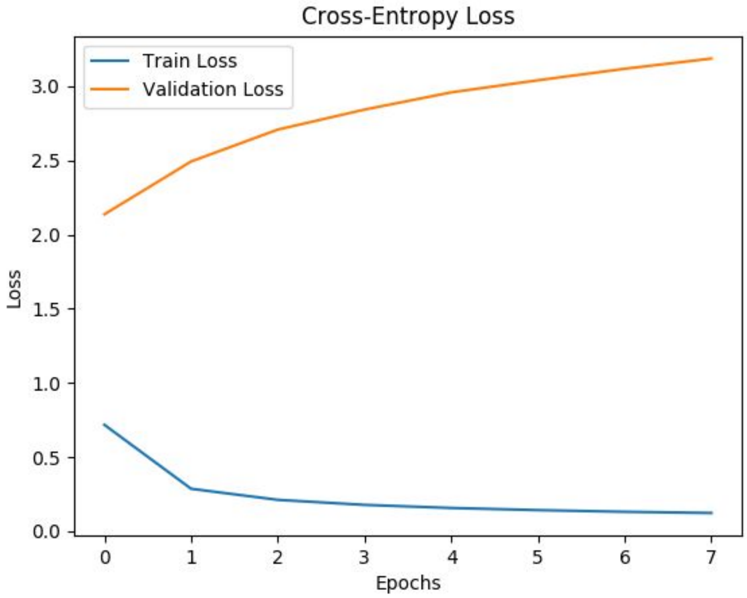




Sequential:

Train Loss	:	0.1233
Cross Validation Loss	:	3.1875
Test Loss	:	2.7415
Train Accuracy	:	99.79 %
Cross Validation Accuracy	:	**40.62** % (Improvement over classification: 38.75 %)
Test Accuracy	:	**47.72** % (Improvement over classification: 46.95%)


Model Accuracy

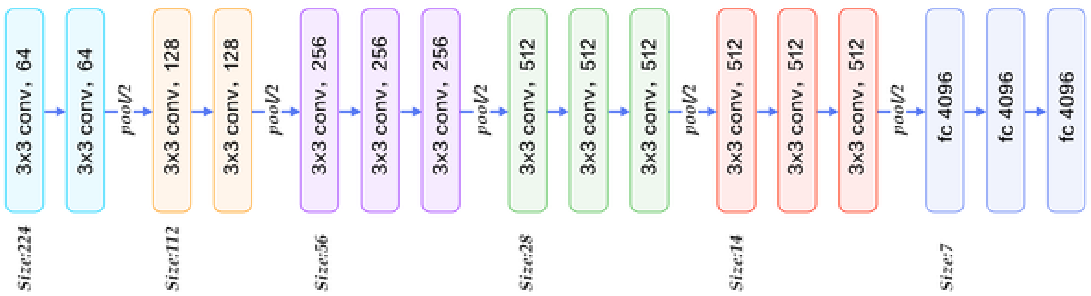

Cross-Entropy Loss

Observation:

- The training accuracy is very high compared to test accuracy. This shows that the network overfits the data.
- This is probably because the network weights are randomly initialized.
- The original paper uses the same network pre-trained on ImageNet dataset.
- Since it is not possible to train the current network on the ImageNet dataset, we opt for different architecture(VGG-16) which is available pre-trained on ImageNet.

Approach 2: Transfer Learning (VGG-16 pretrained model)

The VGG-16 network is shown below:



https://goo.gl/images/1kAGbi

- The network is pretrained on ImageNet Dataset for 1000 classes
- **Only the convolutional part of the network is taken**

Classification Model:

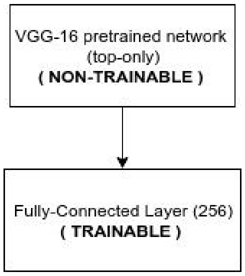
- The **Convolutional part** is made **non-trainable** so it acts as feature extractor
- A fully connected layer is added after flattening with 256 neurons (which is **trainable**)

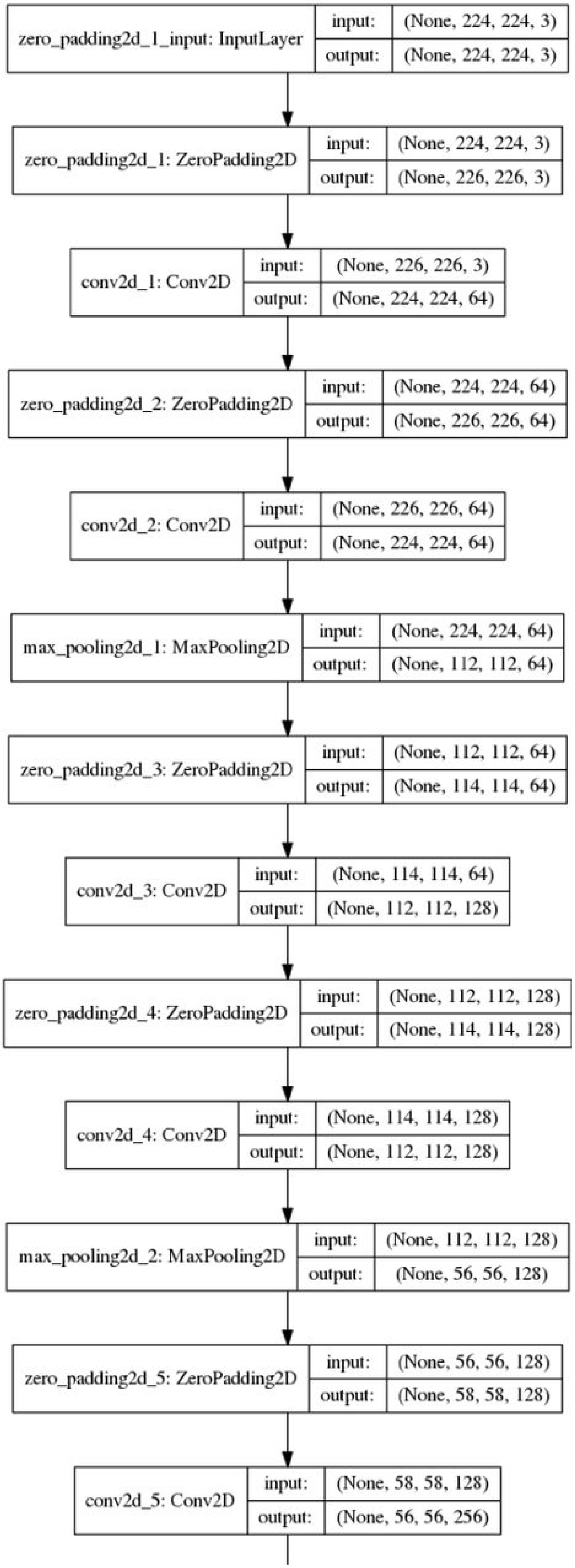
- Classification: Softmax
- Loss: Categorical Cross Entropy
- Optimizer: Adam

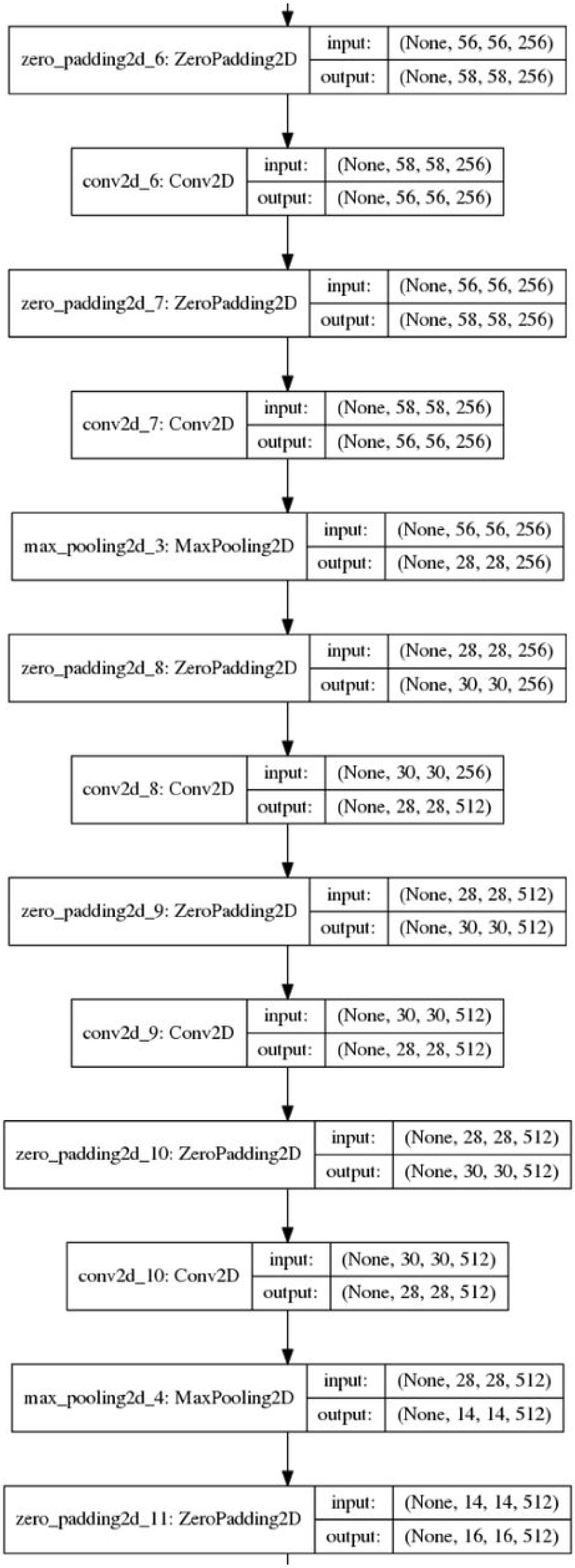
- Total parameters: 21,139,271
- Total Trainable parameters: 6,424,583

Hyper-parameters:

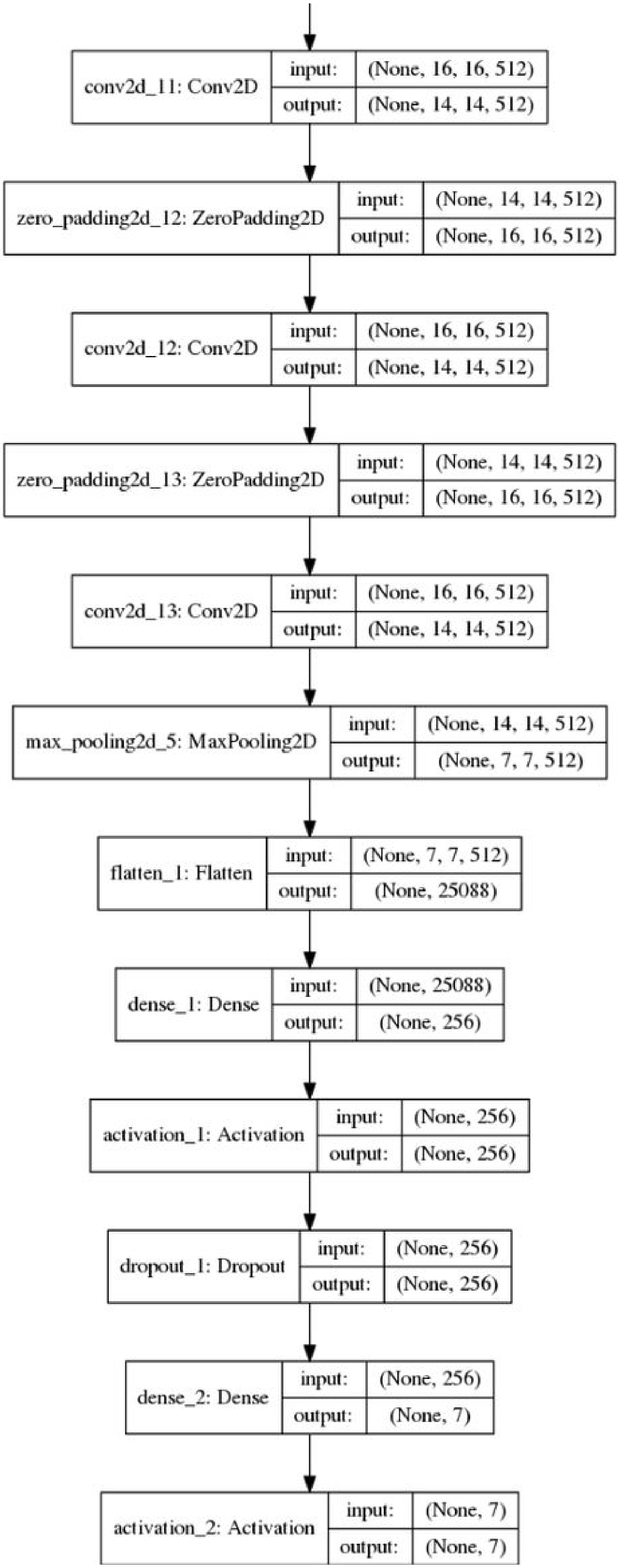
- Regularization: Dropout (probability: 0.5)
- Batch size: 128
- Learning Rate: 1e-4
- Decay Rate: 1e-2




zero_padding2d_1_input: InputLayer
input:
(None, 224, 224, 3)
output:
(None, 224, 224, 3)
zero_padding2d_1: ZeroPadding2D
input:
(None, 224, 224, 3)
output:
(None, 226, 226, 3)
conv2d_1: Conv2D
input:
(None, 226, 226, 3)
output:
(None, 224, 224, 64)
zero_padding2d_2: ZeroPadding2D
input:
(None, 224, 224, 64)
output:
(None, 226, 226, 64)
conv2d_2: Conv2D
input:
(None, 226, 226, 64)
output:
(None, 224, 224, 64)
max_pooling2d_1: MaxPooling2D
input:
(None, 224, 224, 64)
output:
(None, 112, 112, 64)
zero_padding2d_3: ZeroPadding2D
input:
(None, 112, 112, 64)
output:
(None, 114, 114, 64)
conv2d_3: Conv2D
input:
(None, 114, 114, 64)
output:
(None, 112, 112, 128)
zero_padding2d_4: ZeroPadding2D
input:
(None, 112, 112, 128)
output:
(None, 114, 114, 128)
conv2d_4: Conv2D
input:
(None, 114, 114, 128)
output:
(None, 112, 112, 128)
max_pooling2d_2: MaxPooling2D
input:
(None, 112, 112, 128)
output:
(None, 56, 56, 128)
zero_padding2d_5: ZeroPadding2D
input:
(None, 56, 56, 128)
output:
(None, 58, 58, 128)
conv2d_5: Conv2D
input:
(None, 58, 58, 128)
output:
(None, 56, 56, 256)



zero_padding2d_6: ZeroPadding2D	input:	(None, 56, 56, 256)
	output:	(None, 58, 58, 256)
conv2d_6: Conv2D	input:	(None, 58, 58, 256)
	output:	(None, 56, 56, 256)
zero_padding2d_7: ZeroPadding2D	input:	(None, 56, 56, 256)
	output:	(None, 58, 58, 256)
conv2d_7: Conv2D	input:	(None, 58, 58, 256)
	output:	(None, 56, 56, 256)
max_pooling2d_3: MaxPooling2D	input:	(None, 56, 56, 256)
	output:	(None, 28, 28, 256)
zero_padding2d_8: ZeroPadding2D	input:	(None, 28, 28, 256)
	output:	(None, 30, 30, 256)
conv2d_8: Conv2D	input:	(None, 30, 30, 256)
	output:	(None, 28, 28, 512)
zero_padding2d_9: ZeroPadding2D	input:	(None, 28, 28, 512)
	output:	(None, 30, 30, 512)
conv2d_9: Conv2D	input:	(None, 30, 30, 512)
	output:	(None, 28, 28, 512)
zero_padding2d_10: ZeroPadding2D	input:	(None, 28, 28, 512)
	output:	(None, 30, 30, 512)
conv2d_10: Conv2D	input:	(None, 30, 30, 512)
	output:	(None, 28, 28, 512)
max_pooling2d_4: MaxPooling2D	input:	(None, 28, 28, 512)
	output:	(None, 14, 14, 512)
zero_padding2d_11: ZeroPadding2D	input:	(None, 14, 14, 512)
	output:	(None, 16, 16, 512)


conv2d_11: Conv2D
input: (None, 16, 16, 512)
output: (None, 14, 14, 512)
zero_padding2d_12: ZeroPadding2D
input: (None, 14, 14, 512)
output: (None, 16, 16, 512)
conv2d_12: Conv2D
input: (None, 16, 16, 512)
output: (None, 14, 14, 512)
zero_padding2d_13: ZeroPadding2D
input: (None, 14, 14, 512)
output: (None, 16, 16, 512)
conv2d_13: Conv2D
input: (None, 16, 16, 512)
output: (None, 14, 14, 512)
max_pooling2d_5: MaxPooling2D
input: (None, 14, 14, 512)
output: (None, 7, 7, 512)
flatten_1: Flatten
input: (None, 7, 7, 512)
output: (None, 25088)
dense_1: Dense
input: (None, 25088)
output: (None, 256)
activation_1: Activation
input: (None, 256)
output: (None, 256)
dropout_1: Dropout
input: (None, 256)
output: (None, 256)
dense_2: Dense
input: (None, 256)
output: (None, 7)
activation_2: Activation
input: (None, 7)
output: (None, 7)

Sequential Model:

- The **Convolutional part** and the **fully-connected layer with 256 neurons** is made **non-trainable** so it acts as feature extractor

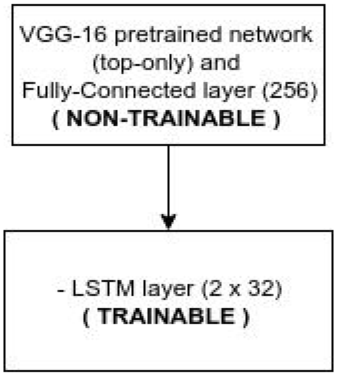
- 2 x LSTM layers
- Classification: Softmax
- Loss: Categorical Cross Entropy
- Optimizer: Adam

- Total parameters: 21,183,015
- Total Trainable parameters: 45,543

Hyper-parameters:

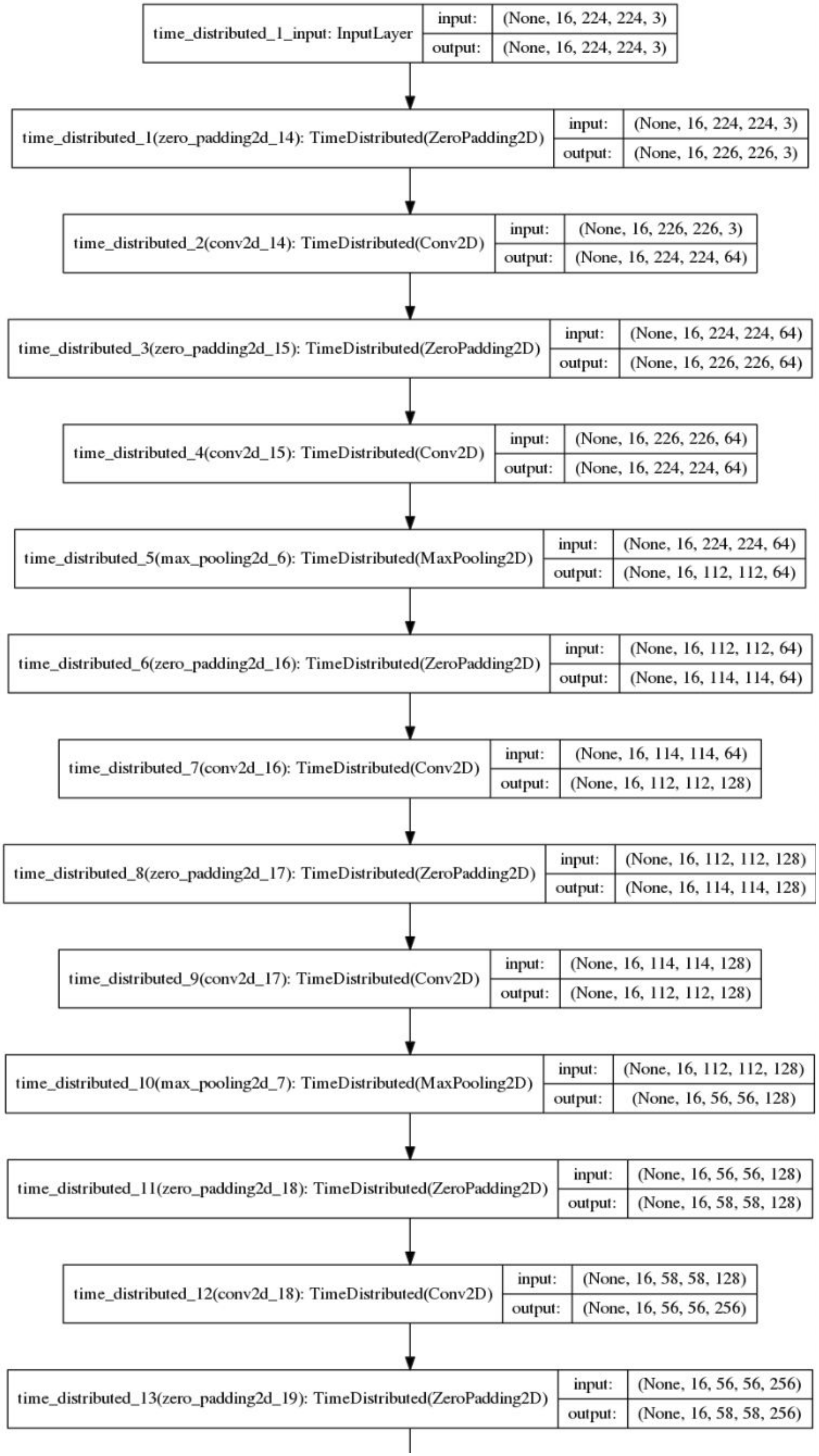
- LSTM cells : 32 cells x 2 layers
- Regularization: Dropout (probability: 0.5)
- Batch size: 32
- Learning Rate: 1e-4
- Decay Rate: 1e-2

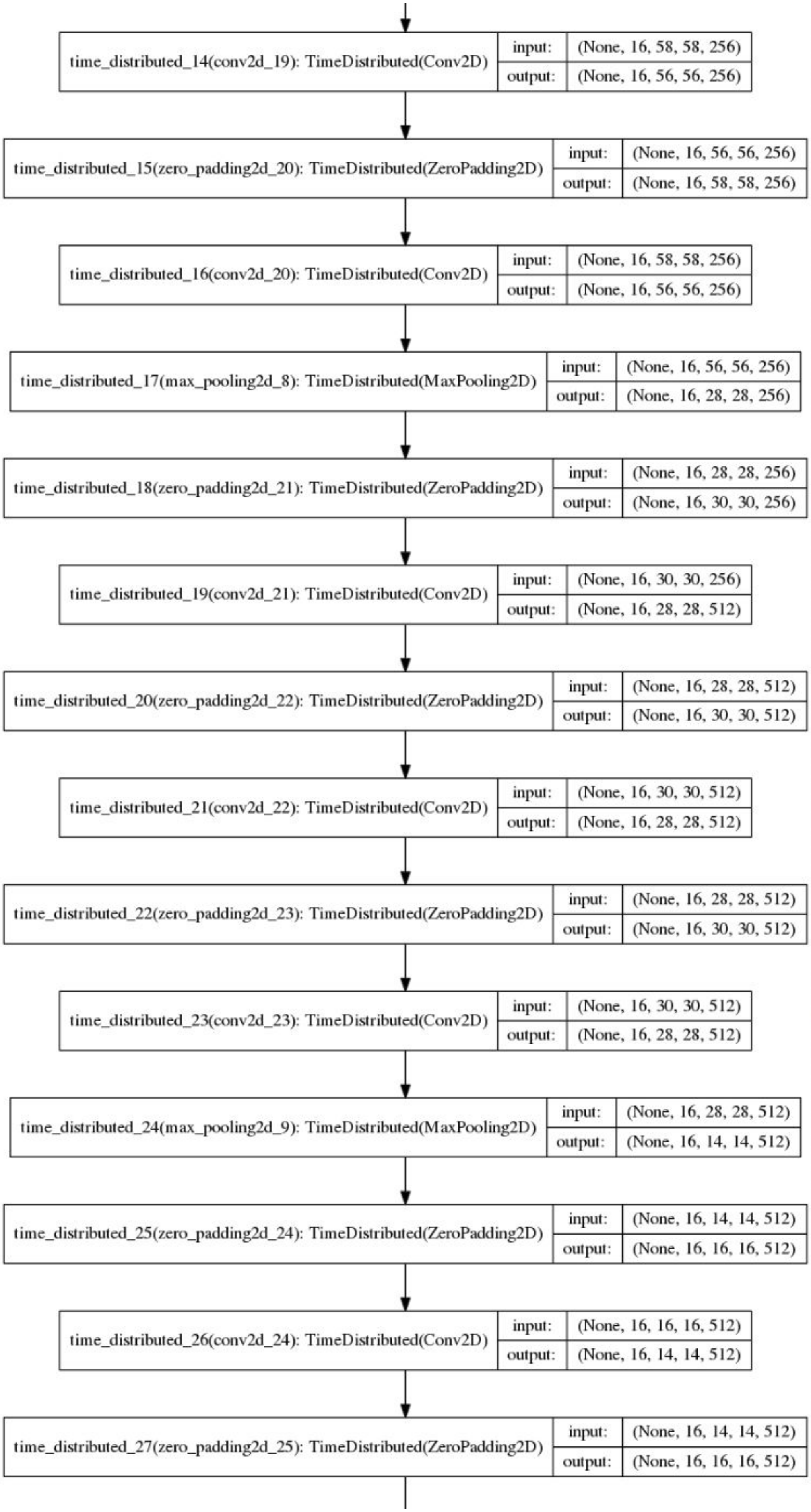
- Due to memory constraint on Henry cluster, batch size is reduced to 32.
- The entire convolutional part with one fully connected layer with 256 neurons are **non-trainable**.
- **Only the LSTM layers are trainable**

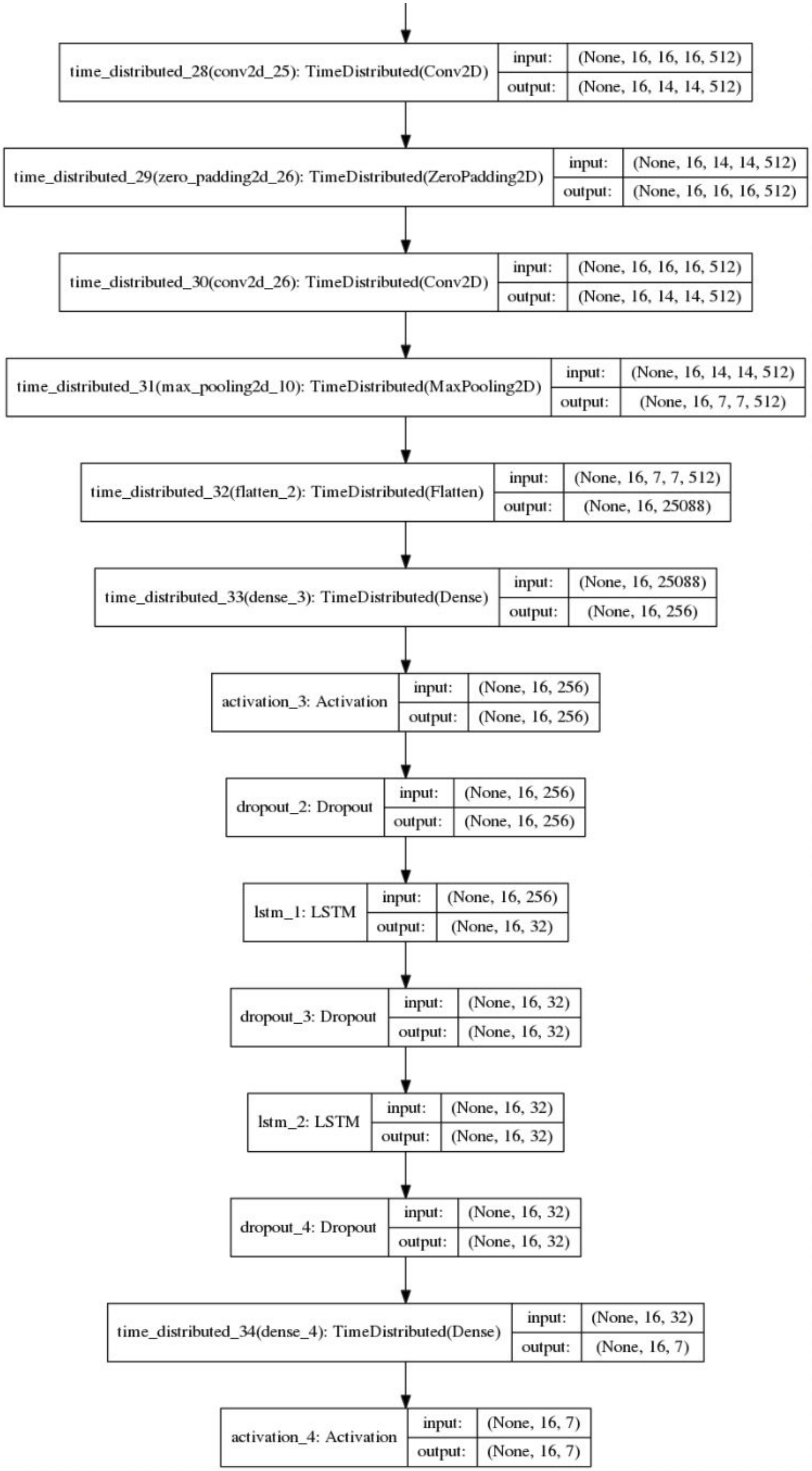


VGG-16 pretrained network (top-only) and Fully-Connected layer (256)
(NON-TRAINABLE)

- LSTM layer (2 x 32)
(TRAINABLE)


time_distributed_1_input: InputLayer
input: (None, 16, 224, 224, 3)
output: (None, 16, 224, 224, 3)
time_distributed_1(zero_padding2d_14): TimeDistributed(ZeroPadding2D)
input: (None, 16, 224, 224, 3)
output: (None, 16, 226, 226, 3)
time_distributed_2(conv2d_14): TimeDistributed(Conv2D)
input: (None, 16, 226, 226, 3)
output: (None, 16, 224, 224, 64)
time_distributed_3(zero_padding2d_15): TimeDistributed(ZeroPadding2D)
input: (None, 16, 224, 224, 64)
output: (None, 16, 226, 226, 64)
time_distributed_4(conv2d_15): TimeDistributed(Conv2D)
input: (None, 16, 226, 226, 64)
output: (None, 16, 224, 224, 64)
time_distributed_5(max_pooling2d_6): TimeDistributed(MaxPooling2D)
input: (None, 16, 224, 224, 64)
output: (None, 16, 112, 112, 64)
time_distributed_6(zero_padding2d_16): TimeDistributed(ZeroPadding2D)
input: (None, 16, 112, 112, 64)
output: (None, 16, 114, 114, 64)
time_distributed_7(conv2d_16): TimeDistributed(Conv2D)
input: (None, 16, 114, 114, 64)
output: (None, 16, 112, 112, 128)
time_distributed_8(zero_padding2d_17): TimeDistributed(ZeroPadding2D)
input: (None, 16, 112, 112, 128)
output: (None, 16, 114, 114, 128)
time_distributed_9(conv2d_17): TimeDistributed(Conv2D)
input: (None, 16, 114, 114, 128)
output: (None, 16, 112, 112, 128)
time_distributed_10(max_pooling2d_7): TimeDistributed(MaxPooling2D)
input: (None, 16, 112, 112, 128)
output: (None, 16, 56, 56, 128)
time_distributed_11(zero_padding2d_18): TimeDistributed(ZeroPadding2D)
input: (None, 16, 56, 56, 128)
output: (None, 16, 58, 58, 128)
time_distributed_12(conv2d_18): TimeDistributed(Conv2D)
input: (None, 16, 58, 58, 128)
output: (None, 16, 56, 56, 256)
time_distributed_13(zero_padding2d_19): TimeDistributed(ZeroPadding2D)
input: (None, 16, 56, 56, 256)
output: (None, 16, 58, 58, 256)


time_distributed_14(conv2d_19): TimeDistributed(Conv2D)
input: (None, 16, 58, 58, 256)
output: (None, 16, 56, 56, 256)
time_distributed_15(zero_padding2d_20): TimeDistributed(ZeroPadding2D)
input: (None, 16, 56, 56, 256)
output: (None, 16, 58, 58, 256)
time_distributed_16(conv2d_20): TimeDistributed(Conv2D)
input: (None, 16, 58, 58, 256)
output: (None, 16, 56, 56, 256)
time_distributed_17(max_pooling2d_8): TimeDistributed(MaxPooling2D)
input: (None, 16, 56, 56, 256)
output: (None, 16, 28, 28, 256)
time_distributed_18(zero_padding2d_21): TimeDistributed(ZeroPadding2D)
input: (None, 16, 28, 28, 256)
output: (None, 16, 30, 30, 256)
time_distributed_19(conv2d_21): TimeDistributed(Conv2D)
input: (None, 16, 30, 30, 256)
output: (None, 16, 28, 28, 512)
time_distributed_20(zero_padding2d_22): TimeDistributed(ZeroPadding2D)
input: (None, 16, 28, 28, 512)
output: (None, 16, 30, 30, 512)
time_distributed_21(conv2d_22): TimeDistributed(Conv2D)
input: (None, 16, 30, 30, 512)
output: (None, 16, 28, 28, 512)
time_distributed_22(zero_padding2d_23): TimeDistributed(ZeroPadding2D)
input: (None, 16, 28, 28, 512)
output: (None, 16, 30, 30, 512)
time_distributed_23(conv2d_23): TimeDistributed(Conv2D)
input: (None, 16, 30, 30, 512)
output: (None, 16, 28, 28, 512)
time_distributed_24(max_pooling2d_9): TimeDistributed(MaxPooling2D)
input: (None, 16, 28, 28, 512)
output: (None, 16, 14, 14, 512)
time_distributed_25(zero_padding2d_24): TimeDistributed(ZeroPadding2D)
input: (None, 16, 14, 14, 512)
output: (None, 16, 16, 16, 512)
time_distributed_26(conv2d_24): TimeDistributed(Conv2D)
input: (None, 16, 16, 16, 512)
output: (None, 16, 14, 14, 512)
time_distributed_27(zero_padding2d_25): TimeDistributed(ZeroPadding2D)
input: (None, 16, 14, 14, 512)
output: (None, 16, 16, 16, 512)

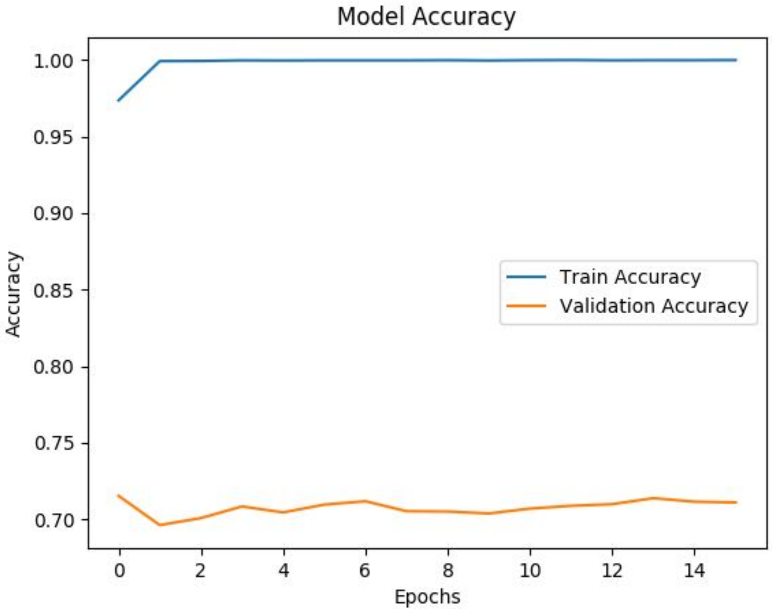

time_distributed_28(conv2d_25): TimeDistributed(Conv2D)
input: (None, 16, 16, 16, 512)
output: (None, 16, 14, 14, 512)
time_distributed_29(zero_padding2d_26): TimeDistributed(ZeroPadding2D)
input: (None, 16, 14, 14, 512)
output: (None, 16, 16, 16, 512)
time_distributed_30(conv2d_26): TimeDistributed(Conv2D)
input: (None, 16, 16, 16, 512)
output: (None, 16, 14, 14, 512)
time_distributed_31(max_pooling2d_10): TimeDistributed(MaxPooling2D)
input: (None, 16, 14, 14, 512)
output: (None, 16, 7, 7, 512)
time_distributed_32(flatten_2): TimeDistributed(Flatten)
input: (None, 16, 7, 7, 512)
output: (None, 16, 25088)
time_distributed_33(dense_3): TimeDistributed(Dense)
input: (None, 16, 25088)
output: (None, 16, 256)
activation_3: Activation
input: (None, 16, 256)
output: (None, 16, 256)
dropout_2: Dropout
input: (None, 16, 256)
output: (None, 16, 256)
lstm_1: LSTM
input: (None, 16, 256)
output: (None, 16, 32)
dropout_3: Dropout
input: (None, 16, 32)
output: (None, 16, 32)
lstm_2: LSTM
input: (None, 16, 32)
output: (None, 16, 32)
dropout_4: Dropout
input: (None, 16, 32)
output: (None, 16, 32)
time_distributed_34(dense_4): TimeDistributed(Dense)
input: (None, 16, 32)
output: (None, 16, 7)
activation_4: Activation
input: (None, 16, 7)
output: (None, 16, 7)

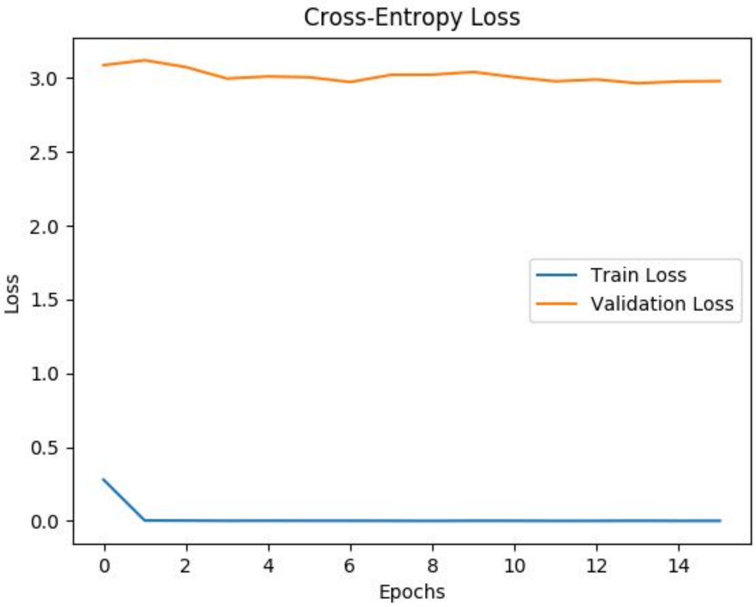
Result:

Classification:

Train Loss : 1.193947e-7
Cross Validation Loss : 2.9791
Test Loss : 4.08048

Train Accuracy : 100.00 %
Cross Validation Accuracy : **71.52** % (Improvement over Approach 1(Sequential) : 40.62%)
Test Accuracy : **60.71** % (Improvement over Approach 1(Sequential) : 47.72%)

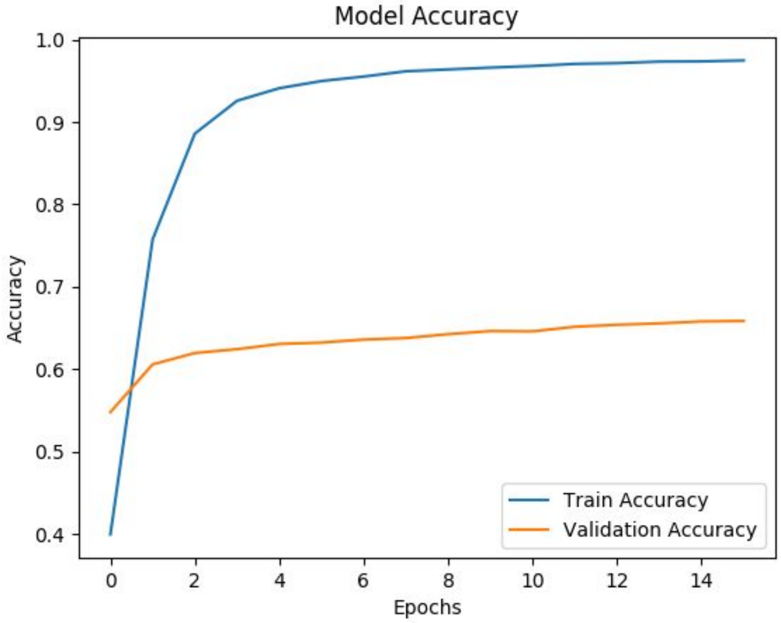


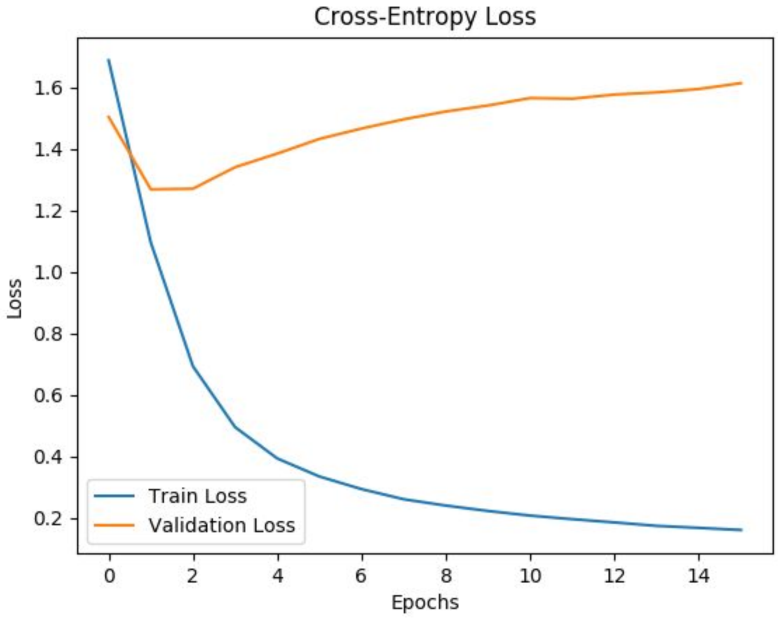


Sequential:

Train Loss	:	0.1603
Cross Validation Loss	:	1.6146
Test Loss	:	1.8774

Train Accuracy	:	97.47 %
Cross Validation Accuracy	:	65.86 % (Results did not improve compared to classification.
Test Accuracy	:	57.78 % Requires Hyper-parameter tuning.)





Code implementation of the report:
https://github.com/suraj-maniyar/Activity-Recognition-From-Video