Model Architecture

June 13, 2024

1 TimeDistributed Layer

1.1 Overview

The ‘TimeDistributed‘ layer in TensorFlow is designed to apply the same operation across each time step of a sequence independently. This layer is particularly useful in processing time series data or sequences of data in recurrent neural networks (RNNs) and other sequence-based models.

1.2 Functionality

The primary functionality of the ‘TimeDistributed‘ layer is to wrap another layer so that the wrapped layer is applied to each time slice or temporal segment of the input data. This is useful when working with 3D or higher-dimensional data where operations need to be applied across the temporal dimension.

- **Input Shape**: The typical input shape is $(batch\_size, time\_steps, ...)$, where ‘time_steps‘ represent the sequence length.
- **Output Shape**: The output shape is usually $(batch\_size, time\_steps, ...)$, where the operations are applied independently across each time step.

1.3 Graphical Representation

2 Practical Applications

2.1 Applying TimeDistributed Layer

When the input is a sequence of images with difference in time stamps like a burst photo in a phone,one way to approach this problem is to merge all the images into a single image and input the merged image to the model but here we will lose key information which can help the model to predict. In our problem statement, Small differences in the location of objects in the images can help the model determine the angular velocity of the satellite but by merging the image we will be losing this information.

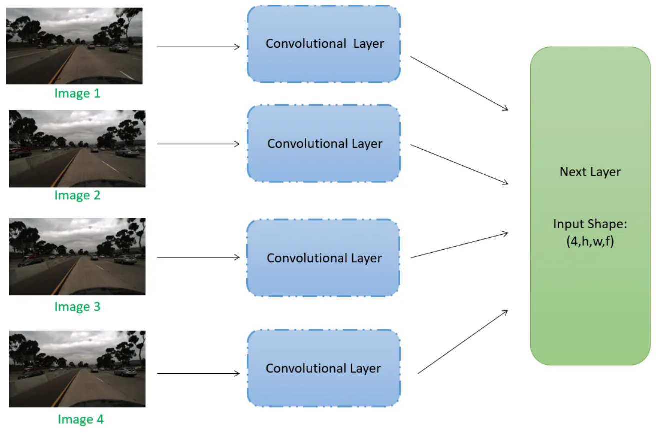

Figure 1: Graphical representation of the ‘TimeDistributed‘ layer applying a ResNet50 layer to each time step of a sequence.

So we have to find a way to not lose information and at the same time pass 10 images to the model and get an output. Here comes our Savior the Time Distributed Layer from Tensorflow.

This Specialized Layer applies the same layer to several inputs and get output for each input such that we can combine them and pass it to another layer to make predictions.

In this way, we use only one layer that performs its operations on 8 separate images and gives output.

The weights of each layer is same. So we are using the same ResNet 50 in our case for feature extraction from each of the 10 images of the sequence.

2.1.1 Model Description

In this example, each time step of the input sequence is processed by the ‘ResNet50‘ layer independently, resulting in an output sequence where each step has been transformed by the ‘ResNet50‘ layer and finally that output flattened 1D vector from each image goes into LSTM for sequence processing .

2.2 Graphical Representation

2.3 Combining TimeDistributed with LSTM

The ‘TimeDistributed‘ layer is often used in combination with recurrent layers like LSTM (Long Short-Term Memory) to process sequences and maintain temporal dependencies.

2.3.1 Example with LSTM and Dense Layers

Here’s how the ‘TimeDistributed‘ layer can be combined with an LSTM layer to process sequential data and then apply a ‘Dense‘ layer to each time step:

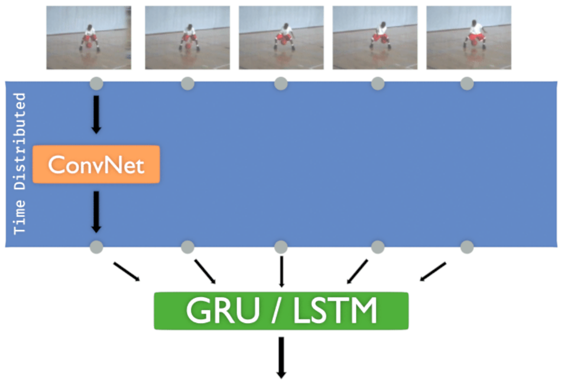

Figure 2: Combining ‘TimeDistributed‘ with LSTM layers for sequence processing.

```
# Define the model
   resnet_base = ResNet50(weights='imagenet', include_top=False, input_shape=(*image_size,
   resnet_model = Sequential([
       resnet_base,
       GlobalAveragePooling2D()
   ])
   time_distributed_resnet = TimeDistributed(resnet_model)

   model = Sequential()
   model.add(time_distributed_resnet)
   model.add(LSTM(256, return_sequences=True))
   model.add(TimeDistributed(Dense(output_size)))
```

In this model, the ‘LSTM‘ layer processes the entire sequence and retains the sequence output, which is then passed to a ‘TimeDistributed‘ ‘Dense‘ layer that applies a dense transformation to each time step.

3 Benefits of Using TimeDistributed Layer

3.1 Simplification of Model Design

Using ‘TimeDistributed‘ layers simplifies the design and implementation of models that need to perform the same operation across different time steps or segments of input data. This is particularly useful for models that process sequences where each element in the sequence requires identical processing steps.

3.2 Flexibility in Handling Variable Length Sequences

The ‘TimeDistributed‘ layer provides flexibility in handling sequences of variable length. By applying the same layer to each time step, the model can process sequences of different lengths without modification to the underlying architecture.

4 Conclusion

Our final Model architecture looks somewhat similar to the below shown image:

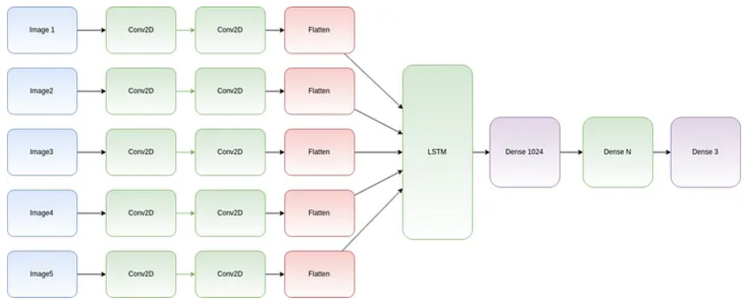

Figure 3: Full Model Architecture.

in our case it takes in a sequence of 10 images as input each of size 224*224*3 and outputs 3 values - yaw,pitch and roll.