**Hand gesture recognition using neural networks**

**Problem Statement:**

Imagine you are working as a data scientist at a home electronics company which manufactures state of the art smart televisions. You want to develop a cool feature in the smart-TV that can recognise five different gestures performed by the user which will help users control the TV without using a remote.

The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

- Thumbs up: Increase the volume

- Thumbs down: Decrease the volume

- Left swipe: 'Jump' backwards 10 seconds

- Right swipe: 'Jump' forward 10 seconds

- Stop: Pause the movie

Each video is a sequence of 30 frames (or images)

**Understanding the Dataset:**

The training data consists of a few hundred videos categorised into one of the five classes. Each video (typically 2-3 seconds long) is divided into a sequence of 30 frames (images). These videos have been recorded by various people performing one of the five gestures in front of a webcam - similar to what the smart TV will use.

**Write-up:**

* **Model 1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| Conv3D | 120x120 | 10, 20 | 0.9154 | 0.8700 |

***Generator function:***

All the 30 images of a video were used for this model. We resized the image to 120x120 and normalized the image using mean and standard deviation.

***Architecture:***

We used 4 Conv3D layers having 8, 16, 32, 64 number of filter having kernel size as (3x3x3) respectively. Batch normalization was applied after each layer convolution layer, followed by a ‘relu’ activation. There were 3 Dense layers having 256, 128, 5 units respectively. We also used dropout for the first two dense layers having rate as 0.25. Activation function used for the first two dense layers were ‘relu’ and ‘softmax’ as the activation function for the last layer. Adam optimization was used as the optimizer.

***End notes:***

The above gave us a pretty good accuracy which was to a satisfactory level. But we thought of to experiment with the input image by flipping, rotating or translating.

* **Model 2: Using left-to-right flip**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| Conv3D | 120x120 | 10, 20 | 0.6343 | 0.5500 |

***Generator function:***

All the 30 images of a video were used for this model. We resized the image to 120x120 and normalized the image using mean and standard deviation. We tried flipping the image left-to-right and likewise we also changed the label as flipping a right-to-left gesture would make it left-to-right.

***Architecture:***

We used 4 Conv3D layers having 8, 16, 32, 64 number of filter having kernel size as (3x3x3) respectively. Batch normalization was applied after each layer convolution layer, followed by a ‘relu’ activation. There were 3 Dense layers having 256, 128, 5 units respectively. We also used dropout for the first two dense layers having rate as 0.25. Activation function used for the first two dense layers were ‘relu’ and ‘softmax’ as the activation function for the last layer. Adam optimization was used as the optimizer.

***End notes:***

The accuracy dropped significantly. Clearly flipping the images was not helping. We still thought to experiment with the images and we tried to use translation and rotation as well in the next model.

* **Model 3: Using left-to-right flip + Translation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| Conv3D | 120x120 | 10, 20 | 0.6501 | 0.7900 |

***Generator function:***

All the 30 images of a video were used for this model. We resized the image to 120x120 and normalized the image using mean and standard deviation. We tried flipping the image left-to-right and likewise we also changed the label as flipping a right-to-left gesture would make it left-to-right. Along with flipping we translated the image.

***Architecture:***

We used 4 Conv3D layers having 8, 16, 32, 64 number of filter having kernel size as (3x3x3) respectively. Batch normalization was applied after each layer convolution layer, followed by a ‘relu’ activation. There were 3 Dense layers having 256, 128, 5 units respectively. We also used dropout for the first two dense layers having rate as 0.25. Activation function used for the first two dense layers were ‘relu’ and ‘softmax’ as the activation function for the last layer. Adam optimization was used as the optimizer.

***End notes:***

This model performed a bit better than Model 2 but still lacked the terms of accuracy compared to Model 1.

* **Model 4: Using left-to-right flip + Rotation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| Conv3D | 120x120 | 10, 20 | 0.6766 | 0.8800 |

***Generator function:***

All the 30 images of a video were used for this model. We resized the image to 120x120 and normalized the image using mean and standard deviation. We tried flipping the image left-to-right and likewise we also changed the label as flipping a right-to-left gesture would make it left-to-right. Along with flipping we tried rotating the image.

***Architecture:***

We used 4 Conv3D layers having 8, 16, 32, 64 number of filter having kernel size as (3x3x3) respectively. Batch normalization was applied after each layer convolution layer, followed by a ‘relu’ activation. There were 3 Dense layers having 256, 128, 5 units respectively. We also used dropout for the first two dense layers having rate as 0.25. Activation function used for the first two dense layers were ‘relu’ and ‘softmax’ as the activation function for the last layer. Adam optimization was used as the optimizer.

***End notes:***

This model performed a bit better than Model 3 but still lacked the terms of accuracy compared to Model 1 in terms of learning from the training set. From model 2, 3 and 4 we came to know that doing any kind of transformation on the sequence does not yield good results, since the gestures performed are for a very short amount of time, given a few seconds in which there is very little tendency of any transformation effect.

* **Model 5:**

Here we have used the same Model 1 from before, the only difference is that we didn’t add dropout after the dense layers to see how that impact the model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| Conv3D | 120x120 | 10, 20 | 0.9652 | 0.8500 |

***Generator function:***

All the 30 images of a video were used for this model. We resized the image to 120x120 and normalized the image using mean and standard deviation.

***Architecture:***

We used 4 Conv3D layers having 8, 16, 32, 64 number of filter having kernel size as (3x3x3) respectively. Batch normalization was applied after each layer convolution layer, followed by a ‘relu’ activation. There were 3 Dense layers having 256, 128, 5 units respectively. Activation function used for the first two dense layers were ‘relu’ and ‘softmax’ as the activation function for the last layer. Adam optimization was used as the optimizer.

***End notes:***

There was an increase in the train accuracy but the test accuracy dropped by 2%. Overall, the performance of the model was satisfactory. Not using dropout clearly could indicate that the model was trying to over-fit the training set and there was almost 11% difference between the train and test accuracy.

Using all the 30 images seemed to be adding a lot of redundant images to the model as there were very little changes in the consecutive images in the video. Hence, we thought of using only 15 images and feed that to the network.

* **Model 6:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| Conv3D | 120x120 | 10, 20 | 0.9303 | 0.8700 |
| 120x120 | 32, 20 | 0.9468 | 0.1000 |
| 120x120 | 64, 20 | 0.8930 | 0.8750 |
| 120x120 | 64, 30 | 0.8128 | 0.1000 |

***Generator function:***

Here we have used 15 images from the video for this model. We resized the image to 120x120 and normalized the image using mean and standard deviation.

***Architecture:***

We used 3 Conv3D layers having 8, 16, 32 number of filter having kernel size as (3x3x3) respectively. Batch normalization was applied after each layer convolution layer, followed by a ‘relu’ activation. There were 3 Dense layers having 256, 128, 5 units respectively. We also used dropout for the first two dense layers having rate as 0.25. Activation function used for the first two dense layers were ‘relu’ and ‘softmax’ as the activation function for the last layer. Adam optimization was used as the optimizer.

***End notes:***

Using only 15 images also gave us better results than using all the 30 images. Since consecutive images does not have a drastic change, they are very similar and you can see movement of hand as you skip some frames and let the model learn from those images. Reducing the number of images also makes the model to train fast.

Also, we tried using different batch sizes and epochs gave us the best results. Using a batch size of 30 for 20 epochs gave us the best accuracy.

Again we thought to experiment and see whether any transformations would result in something different that we got from the previous models.

* **Model 7: Model 6 + Rotation + Translation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| Conv3D | 120x120 | 10, 20 | 0.6169 | 0.5700 |

***Generator function:***

Here we have used 15 images from the video for this model. We resized the image to 120x120 and normalized the image using mean and standard deviation. We tried using rotating and translating the images in the generator function.

***Architecture:***

We used 3 Conv3D layers having 8, 16, 32 number of filter having kernel size as (3x3x3) respectively. Batch normalization was applied after each layer convolution layer, followed by a ‘relu’ activation. There were 3 Dense layers having 256, 128, 5 units respectively. We also used dropout for the first two dense layers having rate as 0.25. Activation function used for the first two dense layers were ‘relu’ and ‘softmax’ as the activation function for the last layer. Adam optimization was used as the optimizer.

***End notes:***

Using rotation + translation on the images was clearly not a good idea.

Since Model 6 gave us very good results, we tried to tweak it a bit. We tried to resize the image to 100x100 and see how it affects the results. We thought of reducing the size further because that would result in less number of trainable parameters and faster learning.

* **Model 8:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| Conv3D | 100x100 | 10, 20 | 0.9552 | 0.9200 |
|  | 100x100 | 32, 20 | 0.9160 | 1.0000 |
|  | 100x100 | 64, 20 | 0.8717 | 0.8750 |
|  | 100x100 | 64, 30 | 0.8235 | 0.7500 |

***Generator function:***

Here we have used 15 images from the video for this model. We resized the image to 100x100 and normalized the image using mean and standard deviation.

***Architecture:***

We used 3 Conv3D layers having 8, 16, 32 number of filter having kernel size as (3x3x3) respectively. Batch normalization was applied after each layer convolution layer, followed by a ‘relu’ activation. There were 3 Dense layers having 256, 128, 5 units respectively. We also used dropout for the first two dense layers having rate as 0.25. Activation function used for the first two dense layers were ‘relu’ and ‘softmax’ as the activation function for the last layer. Adam optimization was used as the optimizer.

***End notes:***

Using only 15 images with 100x100 also gave us good results similar to the Model 6. Also, we tried using different batch sizes and epochs gave us the best results. Using a batch size of 30 for 20 epochs gave us the best accuracy among all the different combinations.

* **Model 9:**

This model is similar to Model 8, the only change is the number of units in the dropout layer is changed from 0.25 to 0.50.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| Conv3D | 100x100 | 32, 20 | 0.6359 | 0.6250 |

***Generator function:***

Here we have used 15 images from the video for this model. We resized the image to 100x100 and normalized the image using mean and standard deviation.

***Architecture:***

We used 3 Conv3D layers having 8, 16, 32 number of filter having kernel size as (3x3x3) respectively. Batch normalization was applied after each layer convolution layer, followed by a ‘relu’ activation. There were 3 Dense layers having 256, 128, 5 units respectively. We also used dropout for the first two dense layers having rate as 0.50. Activation function used for the first two dense layers were ‘relu’ and ‘softmax’ as the activation function for the last layer. Adam optimization was used as the optimizer.

***End notes:***

The accuracy of the model has decreased significantly. Switching off more number of neurons while training had a negative impact on the model as it made it more difficult for the model to learn. Using dropout as 0.25 rather than 0.50 gave us much better results.

* **Model 10:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| VGG16 + GRU | 120x120 | 1, 20 | 0.6244 | 0.6400 |
| 120x120 | 1, 50 | 0.8808 | 0.7200 |

***Generator function:***

Here we have used 15 images from the video for this model. We resized the image to 120x120 and normalized the image using mean and standard deviation.

***Architecture:***

We the pre-trained VGG16 with the layers set as not trainable. Output of the model was a dense layer having 64 units with activation as ‘relu’. After which we used a time-distributed later, followed by 2 GRU layers having 32, 16 units respectively. We also used dropout having rate as 0.25. Follwed that there were two dense layers having 8 units with activation as ‘relu’ and 5 units having activation as ‘softmax’. Adam optimization was used as the optimizer.

***End notes:***

While using batch size as 1 and epochs as 50, after the 32th epoch the test accuracy was plateaued at 0.7000 and didn’t increase any further for around 10 epochs at which point we stopped the training. The accuracy of model was decent.

As we know RNN are known to work really good on sequential data but in this case study we can see that is not always true. Using Conv3D gave us better results and took less time to learn. The number of epochs to get the satisfactory results we far less than compared to using VGG16 + GRU model. We can clearly see that using Conv3D model in this problem statement to recognize gesture are better than using VGG16 + GRU model.

**Summary:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model No.** | **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** | **Description** |
| 1 | Conv3D | 120x120 | 10, 20 | 0.9154 | 0.8700 | Using 30 images. |
| 2 | Conv3D | 120x120 | 10, 20 | 0.6343 | 0.5500 | Using left-to-right flip on the 30 images. |
| 3 | Conv3D | 120x120 | 10, 20 | 0.6501 | 0.7900 | Using left-to-right flip + Translation on the 30 images |
| 4 | Conv3D | 120x120 | 10, 20 | 0.6766 | 0.8800 | Using left-to-right flip + Rotation on the 30 images. |
| 5 | Conv3D | 120x120 | 10, 20 | 0.9652 | 0.8500 | Removing dropout. |
| 6 | Conv3D | 120x120 | 10, 20 | 0.9303 | 0.8700 | Using 15 images rather than 30 images. |
| 120x120 | 32, 20 | 0.9468 | 0.1000 |
| 120x120 | 64, 20 | 0.8930 | 0.8750 |
| 120x120 | 64, 30 | 0.8128 | 0.1000 |
| 7 | Conv3D | 120x120 | 10, 20 | 0.6169 | 0.5700 | Model 6 + Rotation + Translation |
| 8 | Conv3D | 100x100 | 10, 20 | 0.9552 | 0.9200 | Using 100x100 size image. |
| 100x100 | 32, 20 | 0.9160 | 1.0000 |
| 100x100 | 64, 20 | 0.8717 | 0.8750 |
| 100x100 | 64, 30 | 0.8235 | 0.7500 |
| 9 | Conv3D | 100x100 | 32, 20 | 0.6359 | 0.6250 | Changing the number of units in dropout to 0.50 in Model 8. |
| 10 | VGG16 + GRU | 120x120 | 1, 20 | 0.6244 | 0.6400 | VGG16 + GRU and tried for different epochs |
| 120x120 | 1, 50 | 0.8808 | 0.7200 |

**Final Model:**

Using small image size along with fewer layers compared to the other models, Model 8 has shown us satisfactory result and hence we have chosen Model 8 as our final Model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model No.** | **Model** | **Image size** | **Batch size and #epochs** | **Train accuracy** | **Test accuracy** |
| 8 | Conv3D | 100x100 | 10, 20 | 0.9552 | 0.9200 |