#### **Deep Learning Course Project- Gesture Recognition**

#### **Sefali Surabhi Rout – Group Facilitator**

1. **Mudda Rakesh Kumar Reddy**

#### **Problem Statement:**

A model that will be able to predict the 5 gestures correctly is to be built. 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

**Two types of architectures suggested for analysing videos using deep learning:**

1. 3D Convolutional Neural Networks (Conv3D)
2. CNN + RNN architecture

#### **3D Convolutional Neural Networks (Conv3D)**

#### 3D convolutions are a natural extension to the 2D convolutions you are already familiar with. Just like in 2D conv, you move the filter in two directions (x and y), in 3D conv, you move the filter in three directions (x, y and z). In this case, the input to a 3D conv is a video (which is a sequence of 30 RGB images). If we assume that the shape of each image is 100 x 100 x 3, for example, the video becomes a 4D tensor of shape 100 x 100 x 3 x 30 which can be written as (100 x 100 x 30) x 3 where 3 is the number of channels. Hence, deriving the analogy from 2D convolutions where a 2D kernel/filter (a square filter) is represented as (f x f) x c where f is filter size and c is the number of channels, a 3D kernel/filter (a 'cubic' filter) is represented as (f x f x f) x c (here c = 3 since the input images have three channels). This cubic filter will now '3D-convolve' on each of the three channels of the (100 x 100 x 30) tensor.

#### **CNN + RNN architecture**

The conv2D network will extract a feature vector for each image, and a sequence of these feature vectors is then fed to an RNN-based network. The output of the RNN is a regular softmax (for a classification problem such as this one).

**Architecture -1: 3D Convolutional Neural Networks (Conv3D)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Specifications** | **Result (training and validation accuracy)** | **Decision + Explanation** |
| 1 | Conv3D. | Batch size=60 No. of images in img\_idx=30  Image size=64×64  No. of epochs = 8  Trainable params: 395,429 | Train Accuracy:99.85% Validation Accuracy:17% | Conclusion: Overfitting  Decision: Adding dropout layer. |
| 2 | Conv3D. | Batch size=60 No. of images in img\_idx=30  Image size=64×64  No. of epochs = 8  Trainable params: 395,429 | Train Accuracy:83.47% Validation Accuracy:14% | Conclusion: Overfitting Decision: change Batch size,  No. of images in img\_idx,  No. of epochs  and model architecture |
| 3 | Conv3D. | Batch size=30 No. of images in img\_idx=15  Image size=64×64  No. of epochs = 10  Trainable params: 299,301 | Train Accuracy:74.21% Validation Accuracy:26% | Conclusion: Overfitting Decision: change  No. of images in img\_idx,  and model architecture |
| 4 | Conv3D | Batch size=30 No. of images in img\_idx=12  Image size=64×64  No. of epochs = 10  Trainable params: 818,917 | Train Accuracy: 92.61% Validation Accuracy: 27% | Conclusion: Overfitting Decision: change  model architecture |
| 5 | Conv3D | Batch size=30 No. of images in img\_idx=12  Image size=64×64  No. of epochs = 10  Trainable params: 595,813 | Train Accuracy: 100% Validation Accuracy: 23% | Conclusion: Overfitting Decision: add dropout layers and reduce epochs. |
| 6 | Conv3D | Batch size=30 No. of images in img\_idx=12  Image size=64×64  No. of epochs = 5  Trainable params: 595,813 | Train Accuracy: 85.22% Validation Accuracy: 31% | Conclusion: Overfitting Decision: Change model architecture |
| 7 | Conv3D | Batch size=30 No. of images in img\_idx=12  Image size=64×64  No. of epochs = 5  Trainable params: 82,885 | Train Accuracy: 69.38% Validation Accuracy: 16% | Conclusion: Overfitting Decision: add dropout layers |
| 8 | Conv3D | Batch size=30 No. of images in img\_idx=12  Image size=64×64  No. of epochs = 5  Trainable params: 82,885 | Train Accuracy: 42.23% Validation Accuracy: 21% | Conclusion: Overfitting Decision: Change model architecture, batch size and image size |
| 9 | Conv3D | Batch size=25 No. of images in img\_idx=12  Image size=32×32  No. of epochs = 5  Trainable params: 295,813 | Train Accuracy: 94.57% Validation Accuracy: 29% | Conclusion: Overfitting Decision: Change model architecture |
| 10 | Conv3D | Batch size=25 No. of images in img\_idx=12  Image size=32×32  No. of epochs = 5  Trainable params: 132,741 | Train Accuracy: 91.70% Validation Accuracy: 16% | Conclusion: Overfitting Decision: Change model architecture |
| 11 | Conv3D | Batch size=25 No. of images in img\_idx=12  Image size=32×32  No. of epochs = 5  Trainable params: 8733 | Train Accuracy: 45.10% Validation Accuracy: 41% | Conclusion: Not overfitting but poor performance Decision: Change model architecture |
| 12 | Conv3D | Batch size=25 No. of images in img\_idx=12  Image size=32×32  No. of epochs = 5  Trainable params: 20,869 | Train Accuracy: 30.02% Validation Accuracy: 25% | Conclusion: Not overfitting but poor performance |

#### **Architecture - 2: Conv2D + RNN**

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result (training and validation accuracy)** | **Decision + Explanation** |
| **1** | **CNN + GRU** | **90% and 60%** | **Overfit model**  **Trying adding drop outs**  **Trainable params:** 99,269 |
| **2** | **CNN + GRU** | **49% and 41%** | **Resolved overfit**  **Trainable params:** 42,842,565 |
| **3** | **CNN + GRU** | **Batch size and epochs 5, 20**  **55% and 45%** | **Increase the amount of trainable data/ reduce the filter size** |
| **4** | **CNN + GRU Augmentation** | **25% and 29%** |  |
| **5** | **CNN+ RNN+ GRU** | **69% and 61%** | **by increasing epoch and batch size got a model** |
| **6** | **CNN+ RNN+ GRU** | **73% and 62%** | **By layer changes and with less epoch and batch size got a model** |
| **7** | **CNN+ RNN+ LSTM** | **30% and 34%** | **Trainable params:** 57,116,869  **If we increase and batch size and epoch it will better model than GRU but model becomes complex and trainable params are increased** |

#### **model-00020-0.58814-0.72941-0.93977-0.60000.h5** and **model-00050-0.68615-0.68261-1.11482-0.60833.h5** has best model out of all trained models

#### Training Accuracy : 73%, Validation Accuracy : 62% and second model has Training Accuracy : 69%, Validation Accuracy : 61%