Sign Recognition – Machine Learning Project

**Data Preprocessing: (Applies to all Models below)**

1. Resize frames to have 256 diagonal size pixels on the bounding box
2. Randomly crop 224x224 batch from an input frame
3. Apply horizontal flipping by probability of 0.5
4. Extract 50 random frames from each video (we can try out some other ways)
5. Evaluation Metrics: using mean scores of top-k classification accuracies with k={1,5,10} over all the sign instances to help predict the right word between ambiguous words with nearly similar sign language
6. **Image Appearance Based Baselines**
7. **CNN+RNN Model:**
8. **Frames Extraction:**
9. **Fixed number of frames:**

- Pros:

* 1. Simplifies training
  2. **Reduced Computational Cost**

- Cons:

1. Loss of temporal information
2. Bias in frame selection (take one leave one)
3. **All frames of the videos (varying frames):**

- Pros:

* 1. Preserves temporal details
  2. No frame loss

- Cons:

1. Variable sequence length (complicate batching during training)
2. Computational cost
3. **Features extraction:**
4. CNN (VGG16 used in the paper)
5. **Divide into subsets:**
6. Temporal Complexity:
   1. Calculate the difference between consecutive frame features extracted by CNN using this formula where:
      1. = the CNN feature for the i-th
      2. = the number of frames
   2. A high variation indicates more complex dynamics
7. Spatial Complexity:
   1. Feature Sparsity (Count how many zeros are in the matrix)

è = Sum of non-zero values in feature vector for example

1. Grouping inputs by complexity:
2. Cluster videos into subsets (can use clustering algorithms such as k-means)
3. **Temporal dependencies:**
4. RNN (GRU): start with simplest approach (1 layer) & start adding layers accordingly
5. **Classification:**
6. SVM: more efficient with high dimensional features (dimension reduction?)
7. Softmax: probability distribution over all classes

A screenshot of a computer

Description automatically generated

1. **3D CNN Model:**

* Better to capture the spatial-temporal information in a video.
* It performs better than CNN+RNN all mixed in one model.

1. **Pose Based Baselines**
2. **Pose-RNN:**
3. **Frames Extraction:**
4. **Fixed number of frames:**

- Pros:

* 1. Simplifies training
  2. **Reduced Computational Cost**

- Cons:

1. Loss of temporal information
2. Bias in frame selection (take one leave one)
3. **All frames of the videos (varying frames):**

- Pros:

* 1. Preserves temporal details
  2. No frame loss

- Cons:

1. Variable sequence length (complicate batching during training)
2. Computational cost
3. **Extracting 2D Keypoints:**

- Using OpenPose Library

- Extract 13 keypoints upper body, 21 left hand and 21 right hand

1. **RNN:**

- Feed the keypoints to the RNN

1. **Pose TGNN**
2. **Frames Extraction:**
3. **Fixed number of frames:**

- Pros:

1. Simplifies training
2. **Reduced Computational Cost**

- Cons:

1. Loss of temporal information
2. Bias in frame selection (take one leave one)
3. **All frames of the videos (varying frames):**

- Pros:

* 1. Preserves temporal details
  2. No frame loss

- Cons:

1. Variable sequence length (complicate batching during training)
2. Computational cost
3. **Extracting 2D Keypoints:**

- Using OpenPose Library

- Extract 13 key points upper body, 21 left hand and 21 right hand

- Represent the human body as a fully connected graph to keep track of temporal dependencies by observing the variation of the key points

1. **Residual Graph Convolutional Block:**
2. Connect the input to some hidden layers
3. The graph represents the human body where the nodes are the key points (joints)
4. This helps preserve important features and enables deeper stacking of layers
5. **Temporal graph Convolutional Network (TGCN):**
6. Feature Extraction Over Time:
   1. Spatial information: captured by the graph convolutional layers (relationships between key points (joints))
   2. Temporal information: captured by stacking the residual blocks (features of each frame) to analyze how key points changed over time
7. Average Pooling:
   1. To reduce the temporal dimension to a fixed-size representation
   2. Average Pooling is applied along the temporal dimension (Initially we have [# of frames x Size of feature vector] matrix due to the feature extraction over time applied before 🡪 after average pooling, the result will be the same size as the feature vector)
8. **Classification:**
   * 1. SVM
     2. Softmax

**Notes:**

- more samples/gloss 🡪better accuracy