INFOMCV Assignment 5 Report

Alejandro Magarzo – Alimohamed Jaffer (38)

I. Architecture Choices

1. **Frame CNN architecture:** We experimented with many models such as AlexNet, EfficientNet-B0, MobileNetV2 but ultimately for both Stanford and HMDB we used the Resnet-18 model as it was the best performing model. We begin with an input size (3,224 , 224). The first convolutional layer has 64 filters of size 7x7, stride 2 thus, this helps in reducing the size of the input image when capturing the initial features. This results in a feature map of size 112x112. The model has a batch normalization (BatchNorm2d-2) which stabilizes the learning by normalizing the input layers. The reason why it is used after the first convolution layer is because it aids in speeding up the convergence and improving the overall efficiency. We also have a ReLU activation function to introduce non-linearity and thus, allowing the network to capture more complex patterns. Thereafter, it is followed by max pooling of a 3x3 kernel size and stride 2. This is to reduce the spatial size of the representation so as to decrease the computational complexity and the number of parameters to help in preventing overfitting. This will result in an output shape of 56x56. In the final layer, there is an adaptive average pooling layer that is applied which reduces each feature channel to a single scalar value and thus, output shape is a 1x1. There is also a fully connected layer (Linear-68) which reduces the features to a final classification. We used 30 epochs, a learning rate of 0.003, Adam optimizer, and we used the cross entropy loss function.

2. **Optical flow CNN architecture**: For the optical flow, we used the same model as for the frames but with one small modification. Firstly, the input size was changed to (48, 224, 224) and this was because for the optical flow we wanted to have a stack of 16 optical flow frames together. This means that we have to store 16 frames per label. Each frame was considered as an RGB image and thus, each frame would represent three channels. Therefore, stacking 16 of these frames together results in 16*3 = 48 channels. The first convolutional layer was changed to accommodate for a greater number of input channels and so that model can process the entire depth of the input data.

3. **Combination into two-stream network:** For the two stream network architecture we used the spatial stream where we utilized a pre-trained Resnet-18 model and temporal stream where we used a regular Resnet-18 model.

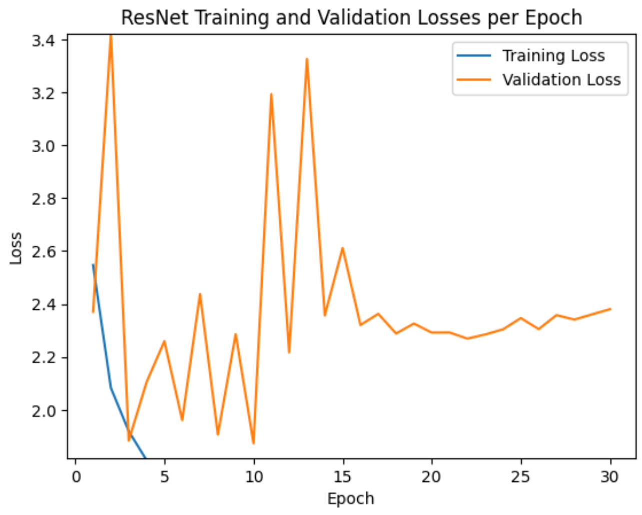
   For our fusion mechanism we first used a late fusion. In this implementation, the outputs from the frames model and the optical model were first processed independently through their respective networks. Thereafter, based on the resulting feature vectors from each network, we then concatenated them. After concatenation, the combined feature vector was passed through a fully connected layer. Therefore, this layer integrates the features from both streams to allow the model to make the final classification.

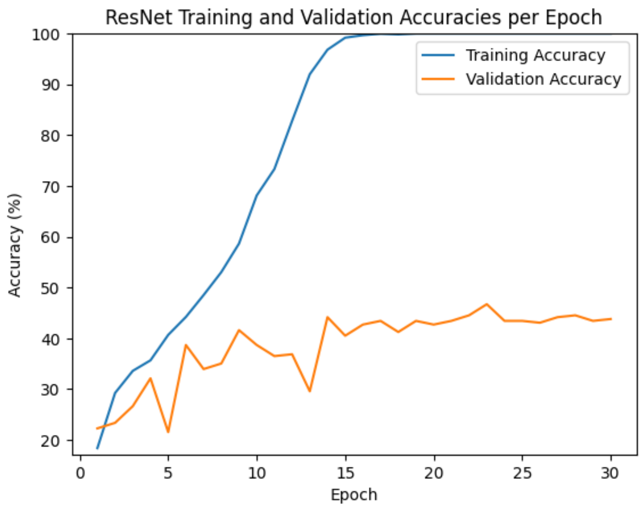
   The decision to use late fusion over intermediate fusion was because late fusion allows for the network to maintain separate processing paths for the spatial and temporal features. This as a result preserves the distinct features until the final stages of the network and we believe that it important to process the data in this way especially for action-recognition tasks.

II. Results

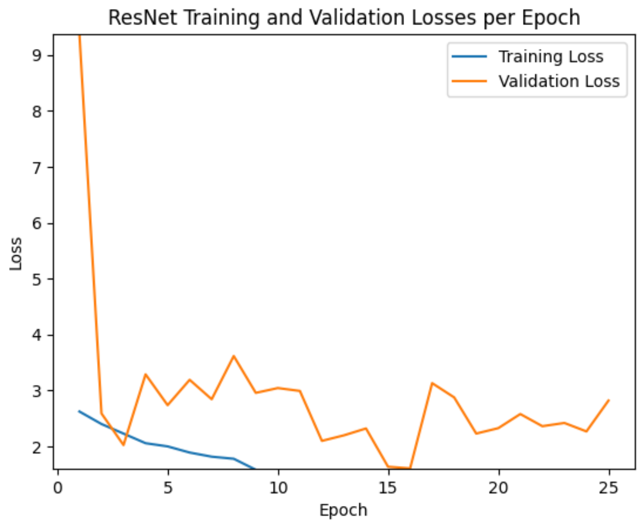
Include the (1) train-validation accuracy and (2) train-validation loss graphs for each of the four networks. Make sure train and validation lines have different colors in your graphs. The x-axis should cover the epochs, the y-axis should be accuracy (for (1)) or loss (for (2)). (two pages)

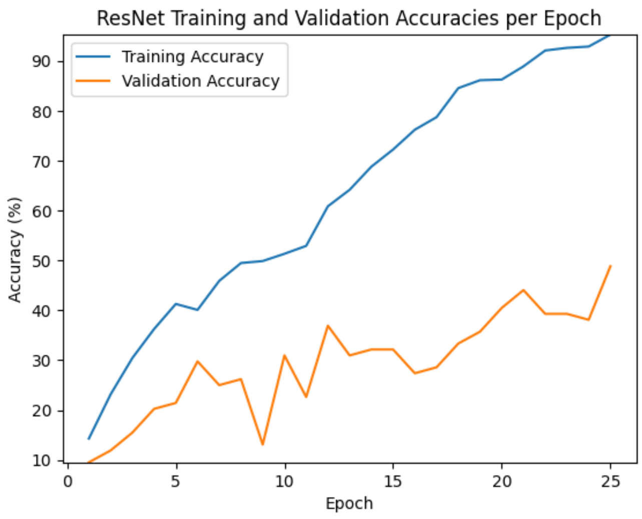
A. Stanford 40 Frames

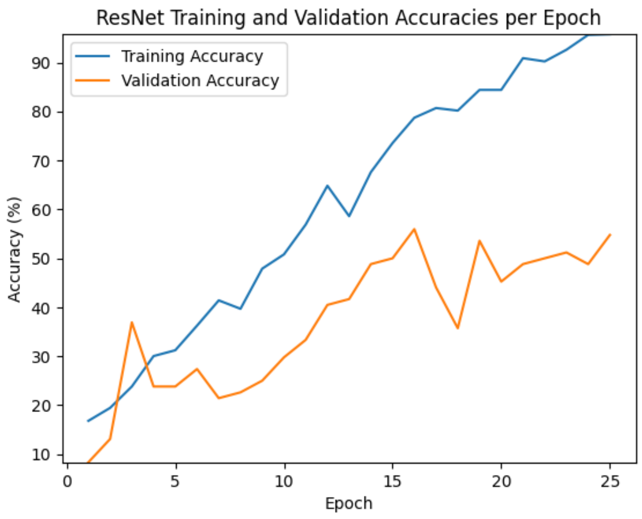




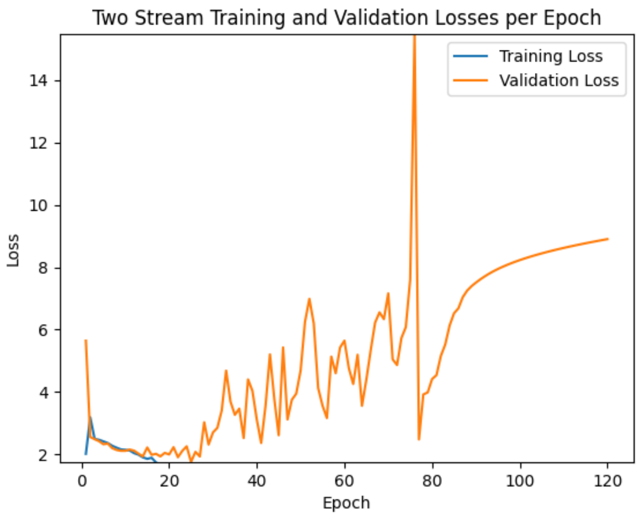
B. HMDB51 Frames

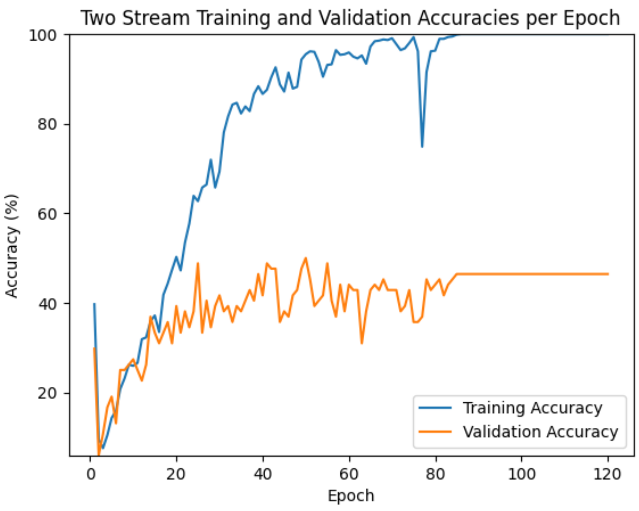




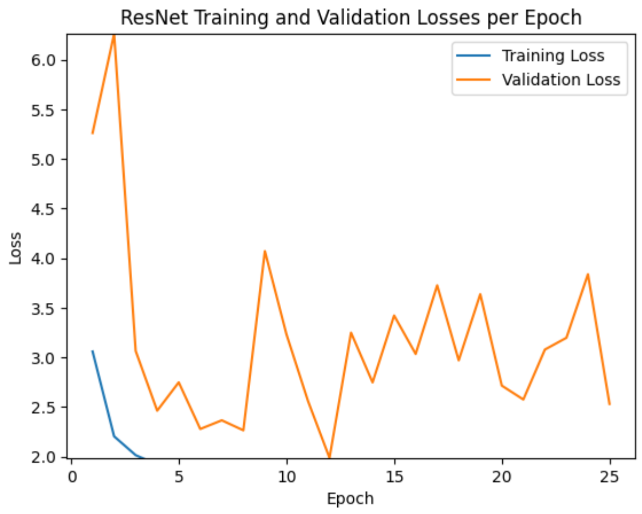


D. HMDB51 Two-stream





C. HMDB51 Optical flow



E. Result table

TABLE I. SUMMARY OF YOUR RESULTS PER MODEL

Dataset	**Model specifications**			
	Model	**Top-1 accuracy (train / test)**	**Top-1 loss train**	**Model parameters**
Stanford 40	Frames	100%/ 42.76%	2.24%	11,689,512
HMDB51	Frames	95.76%/ 30%	2.81%	11,182,668
HMDB51	Optical flow	95.24%/ 25.5%	3.83%	11,830,632
HMDB51	Two-stream	100%/ 46.42%	8.89%	22,406,288

III. Discussion of Results

Stanford-40 Frames: The training accuracy began at 19.32% and reached a final training accuracy of 100%. For the validation score it was variant throughout the 30 epochs as it started at 17.15% and peaked at around 45.62%. Finally the test accuracy was 42.76%. The model showed that there is overfitting and this can be seen by the final training accuracy and final test accuracy scores. This means that the large gap could be due to the model memorizing the training details without truly gaining the ability to generalize well. Furthermore, this can be explained by what we observed in the variations of the validation accuracies throughout the epochs.

HMDB-51 Frames: The training accuracy began at 56.88% and reached a final training accuracy of 95.76%. For the validation score it was variant throughout the 25 epochs as it started at 33.33% and peaked at around 55.95%. Finally the test accuracy was 30%. The model showed that there is overfitting and this can be seen by the final training accuracy and final test accuracy scores. This means that the large gap could be due to the model memorizing the training details without truly gaining the ability to generalize well. Furthermore, this can be explained by what we observed in the variations of the validation accuracies throughout the epochs.

Optical Flow Model: The training accuracy began at 14.29% and reached a final training accuracy of 95.24%. For the validation score it was variant throughout the 25 epochs as it started at 9.52% and peaked at around 48.81%. Finally, the test accuracy was 25.5%. The model showed that there is overfitting and this can be seen by the final training accuracy and final test accuracy scores. This means that the large gap could be due to the model memorizing the training details without truly gaining the ability to generalize well. Furthermore, this can be explained by what we observed in the variations of the validation accuracies throughout the epochs. The overfitting can also be explained that because we are focusing on the motion this therefore brings about complexity of accurately trying to capture and learn from these motion patterns in comparison to static images.

Two-stream Network: The model started with a training accuracy of 39.68% and validation accuracy of 29.76%. The large loss values in the beginning suggest that there was some difficulty in adjusting to the dataset's complexity. By the 120th epoch, the training accuracy achieved 100%, while the validation accuracy peaked around 46.43%. The validation loss also showed significant improvement, stabilizing to a lower value compared to the start. Therefore, the model achieves perfect training accuracy but fails to replicate this performance on the validation set. Furthermore, the test accuracy was 46.42% which proves that there was a lot of overfitting.

All the models showed high training accuracies over time and thus indicating that there was effective learning taking place. However, the models also showed that there was a lot of overfitting due to the disparities in the final training and test accuracy values and the volatile validation accuracies across the epochs.

IV. Link to Model Weights

https://drive.google.com/file/d/1jMmwGSUOOeuE4taxcJXtJKUZ1VgD5qLu/view?usp=sharing

V. Choice Tasks

Choice 1, 2: Data Augmentation for Stanford-40 and HMDB50

Data augmentation was implemented on the Stanford-40 to improve the performance of the model. The model’s test accuracy without data augmentation was 42.76% and after implementing data augmentation the test accuracy increased to 45.72%. The test accuracy for HMDB without applying data augmentation was at 30% and after applying data augmentation, the test accuracy increased to 41.38%.

The first step was to duplicate the original dataset and apply several transformations that would add some variations to the images to create new training examples. The first transformation that was applied was *resize* because we had to ensure that the images were all of uniform size of 256x256 pixels before we apply the other transformations. The next transformation was to perform a *random horizontal flip* which basically randomly flips images horizontally and this as a result adds some symmetrical invariance.

The next transformation was applying a *color jitter* which basically adjusts the brightness, contrast, saturation and hue and this introduces more variability in terms of the color properties. Thus, this would help the model to generalize better across different lighting conditions and color distributions. Our parameters were set to brightness=0.15, contrast=0.15, saturation=0.15, hue=0.05. We chose to set it the brightness parameter value at 0.15 because it would provide a noticeable moderate change to the brightness level of the image and furthermore it would potentially mimic the different lighting conditions that would naturally occur in the environment. Thus, at this parameter level it would not severely alter the appearance of the image. The parameter value for contrast was chosen at 0.15 to ensure that there was a moderate effect on the variation level in terms of enhancing/reducing the intensity differences between the colors and shades in the images. Thus, at this parameter level it ensures that the images are not distorted where the details are lost or exaggerated. We chose a saturation parameter level of 0.15 because this would be enough to represent different color conditions without making colors look unnatural or washed out. Lastly, the hue parameter level was set at 0.05

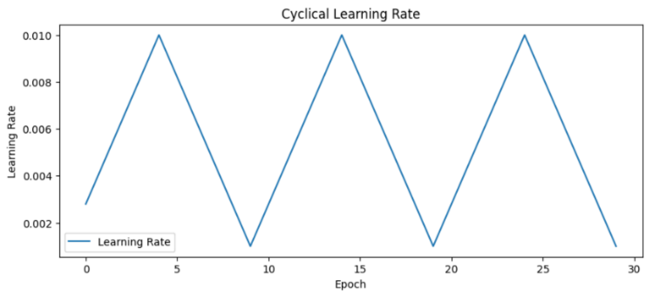
because it would provide a small shift which would mimic the effect of different lighting condition and color temperatures without severely altering the appearance of the image.

The next transformation is to apply a *random rotation* of up to 10 degrees to mimic different camera angles and this is because if the degree of rotation is too much, it would severely alter the appearance of the image. The next transformation is *random resized crop* where we randomly crop images after resizing it and the chosen parameters were size of (224, 224) and scale (0.8, 1.0). The scale chosen has a crop size of between 80% to 100% and this provides a large enough crop where it still contains meaningful parts of the main subjects in the image and thus, avoiding excessive loss of important features. The upper limit of the scale was set to 1.0 to ensure that the entire breadth of the image can be used and thus, this would account for different scenarios where important features are spread out through the image. The last transformation is applying normalization of mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225] which is considered to be the standard values.

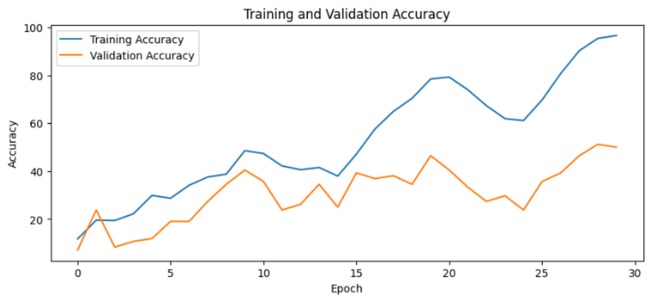
After applying all these transformations to the duplicate dataset, we concatenate the duplicate dataset (with applied transformations) with the original dataset. The goal of this is to increase the size of the dataset and the diversity of the images and thus, has a chance to increase the test accuracy. For both the HMDB and Stanford dataset, the test accuracy increased after applying these data augmentation techniques.

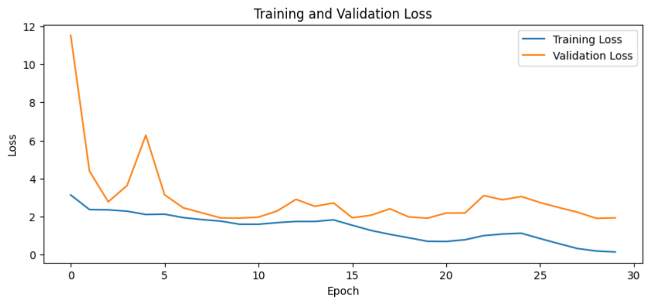
Choice 3: Create a Cyclical Learning rate schedule

The initial and minimum learning rate used was 0.001. The maximum learning rate was set to 0.01. The scheduler should be stepped after every batch we stepped it after epoch. And thus, we set the step size to 5. We set the cyclical momentum to false because in our case it determines whether to adjust the optimizer's momentum in tandem with the learning rate. For our Adam optimizer, momentum adjustments were not necessary.



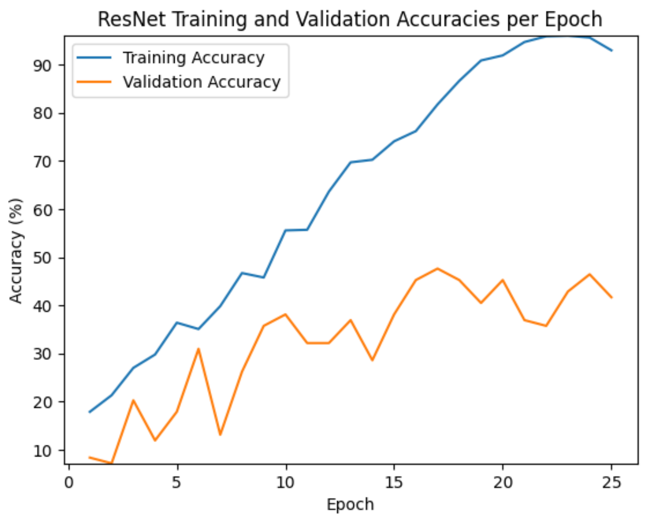
By implementing cyclical rate on the HMDB51 model, the final test accuracy was 26% whereas without implementation of the cyclic rate, the test accuracy was 30%.

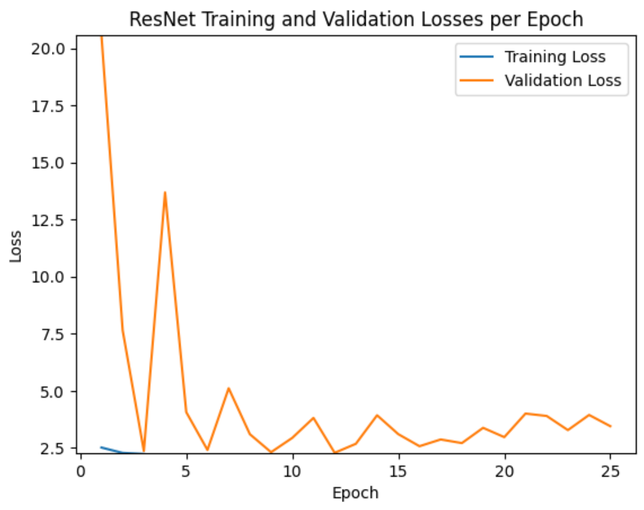




Choice 5: Using First Frame for HMDB50

We attempted to use the first frame instead of the middle frames causes the model to perform worse. This could be because for action recognition, start with a static shot or a title screen that does not represent the action. For example, a video depicting "jumping" might start with the subject standing still before the action begins. The first frame often captures the setup of the scene before any significant movement or action has occurred. This means the model trained on these frames may not learn to recognize the action effectively but instead learns the static context. The middle frames are more likely to capture the peak of the action, where the defining characteristics of the movement are most visible. For action recognition tasks, these frames are crucial as they contain essential visual cues that define the action.



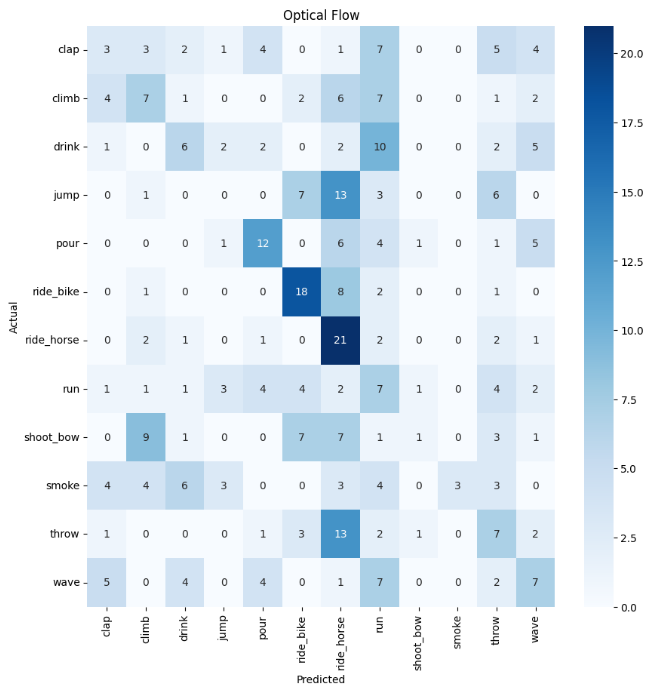


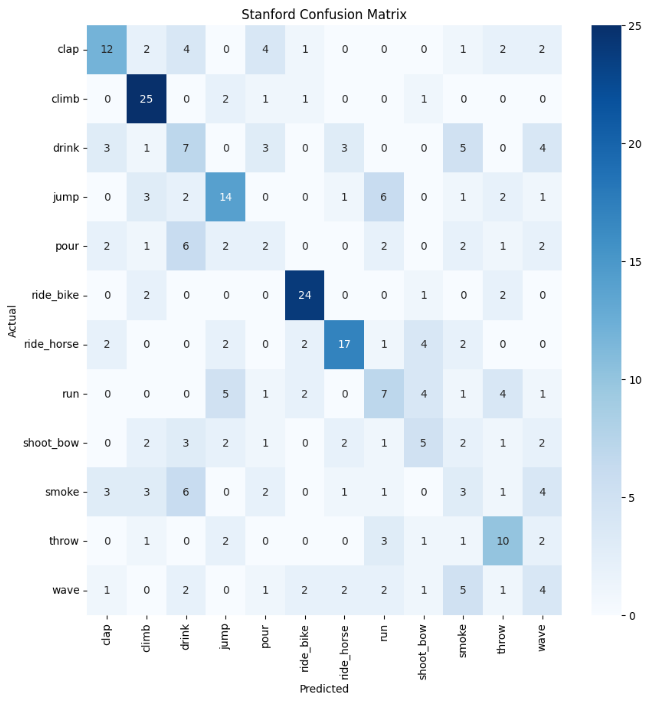
The model had a test accuracy of 22% when using the first frames. We achieved a test accuracy of 30% when using middle frames.

Choice 8: Confusion matrix for Stanford 40 Frames and (2) HMDB51 Optical flow models

The confusion matrix for the Stanford 40 frames showed that “clap” had 12 correct predictions but was also often misclassified as "wave" (10 times). "climb" was mostly correctly predicted (25 times) with few misclassifications. "drink" was often misclassified as "pour" and "wave," indicating possible visual similarities in the actions or model confusion. "ride_bike" and "ride_horse" showed a significant number of correct classifications (24 and 17, respectively). "shoot_bow" showed some confusion with "throw" and "wave," potentially due to similar arm movements. "wave" seemed to be frequently confused with "clap," potentially due to the overlapping nature of hand movement in both actions.

The confusion matrix for the HMDB51 Optical Flow model showed that certain actions like "pour" and "drink", or "clap" and "wave" may have had a similar optical flow patterns leading to confusion. For example, the "drink" action was often misclassified as "pour", which could suggest that the model was catching the similar motion patterns of hand-to-mouth actions but was unable in distinguishing between them well. Also, the matrix suggests that there is some imbalance in the class representation. Some actions like "clap" and "climb" have lower counts, indicating fewer samples in the dataset, which could potentially skew the training. "Ride_bike" showed a good number of correct classifications (15 times) but was also mistaken for "ride_horse" (9 times), which might indicate that the motion patterns of riding were captured but the model could not distinguish between the two different riding contexts. "Ride_horse" had a substantial number of correct predictions (21 times) and was less frequently confused with other actions, indicating that the model captured the motion pattern well. "Run" had 19 correct predictions with a few instances where it is mistaken for "jump" (3 times) and "ride_horse" (2 times), suggesting some confusion with other dynamic actions. "Shoot_bow" was correctly predicted 12 times but showed confusion with "wave" (3 times) and "throw" (3 times), potentially due to the arm extension and pullback movements that are similar in these actions. "Smoke" had 15 correct predictions, which indicated a reasonable ability of the model to recognize this action.


Optical Flow
Actual
Predicted
clap
climb
drink
jump
pour
ride_bike
ride_horse
run
shoot_bow
smoke
throw
wave


Stanford Confusion Matrix
Actual
Predicted
clap
climb
drink
jump
pour
ride_bike
ride_horse
run
shoot_bow
smoke
throw
wave