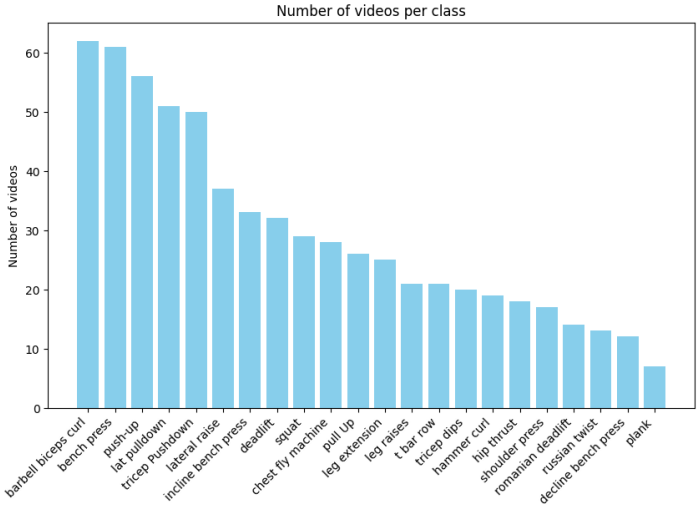
Video Classifier Assignment

1. Data Analysis:

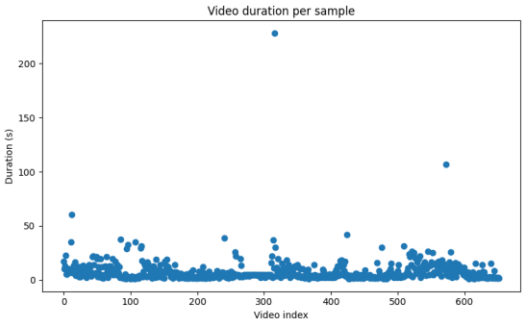
Data analysis is a crucial step in any data-driven project. It involves examining the properties and characteristics of the dataset to understand its structure, identify patterns, and detect potential issues such as missing values, outliers, or imbalanced classes. Overall, thorough data analysis ensures that the model receives clean, well-structured, and meaningful input, which can improve performance and reliability.

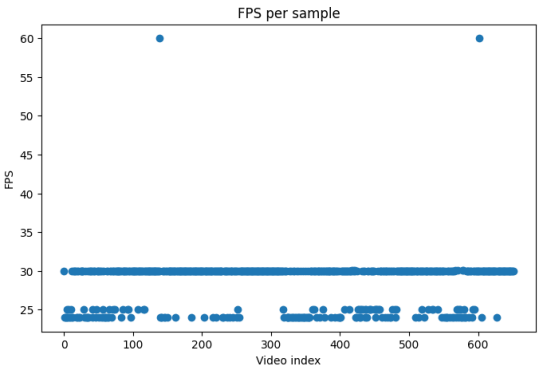
1.1 Number of videos per class:



Since there is a significant imbalance in the number of videos per class (ranging from 7 to 62), this can lead to several issues in the modeling process. Models trained on imbalanced data tend to be biased toward the classes with more samples, which can result in poor performance for the underrepresented classes. To address this, techniques such as oversampling minority classes, applying class-weighted loss, or using data augmentation can be employed to improve performance and ensure fairer predictions across all classes.

1.2 Duration and FPS distribution

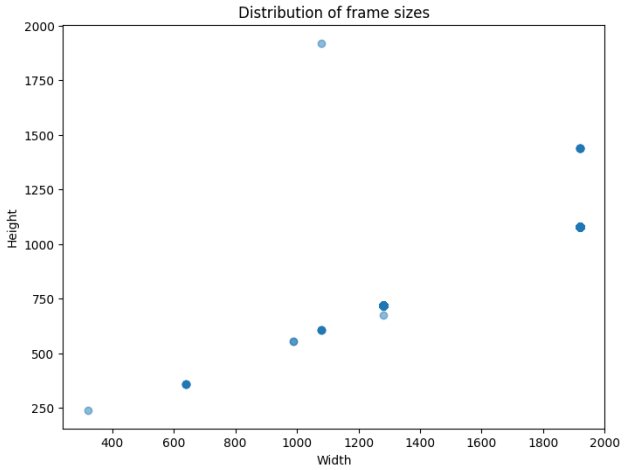




FPS represents the frames per second, indicating how many individual frames (images) are displayed per second in a video. In this dataset, most videos have a similar FPS, ranging from 24 to 30, with only two outliers at higher frame rates. Video durations vary, but most are relatively short, with an average of around 7 seconds and only a few longer outliers. To ensure consistent input, I standardize each video by sampling the same number of frames. This approach preserves the temporal dynamics

for the majority of videos while providing a uniform input size, simplifying feature extraction and model training.

1.3 Distribution of frame sizes



Frame size refers to the spatial dimensions of each video frame. In this dataset, videos have varying frame sizes, meaning some frames are larger or smaller than others. Resizing all frames to a standard resolution ensures spatial consistency, allowing the network to reliably extract features and accurately recognize different exercise movements.

2. Model architecture

To perform the video classification task, I chose to use a **pretrained 3D CNN**. This decision is motivated by the fact that 3D CNNs have a large number of parameters due to both their spatial depth and the temporal dimension (number of frames), which makes training from scratch computationally expensive and time-consuming. Using a pretrained 3D CNN allows me to **fine-tune only a small portion of the weights** (layers 3, 4 and FC) while preserving the model's performance. This approach is particularly suitable for relatively small and imbalanced datasets, where training a deep model end-to-end increases the risk of overfitting. The pretrained network has already learned to extract meaningful spatiotemporal feature representations, so fine-tuning focuses on teaching the model to perform the classification task based on these features. There is a trade-off: training a new model from scratch could potentially discover more task-specific features and achieve higher accuracy, but it would require significantly more time and computational resources. The approach I chose strikes a balance between efficiency and performance. Lower-level layers are kept frozen as they capture generic motion and appearance patterns that are transferable across different video domains, while higher layers are more task-specific.

For this project, I chose the **ResNet R3D-18** architecture as the backbone of the 3D CNN. R3D-18 is a 3D extension of the standard ResNet-18, which applies 3D convolutions over both spatial and temporal dimensions, allowing the network to capture motion and appearance features simultaneously. This network was originally pretrained on the Kinetics dataset, a large-scale action recognition dataset, where it learned to classify a wide range of human actions in videos. This makes it well-suited for classifying workout video, where distinguishing different movements relies not only on the body's spatial posture but also on the temporal patterns of motion.

3. Preprocessing

Preprocessing pipeline includes several steps to ensure the 3D CNN trains effectively and maintains consistency across all video samples:

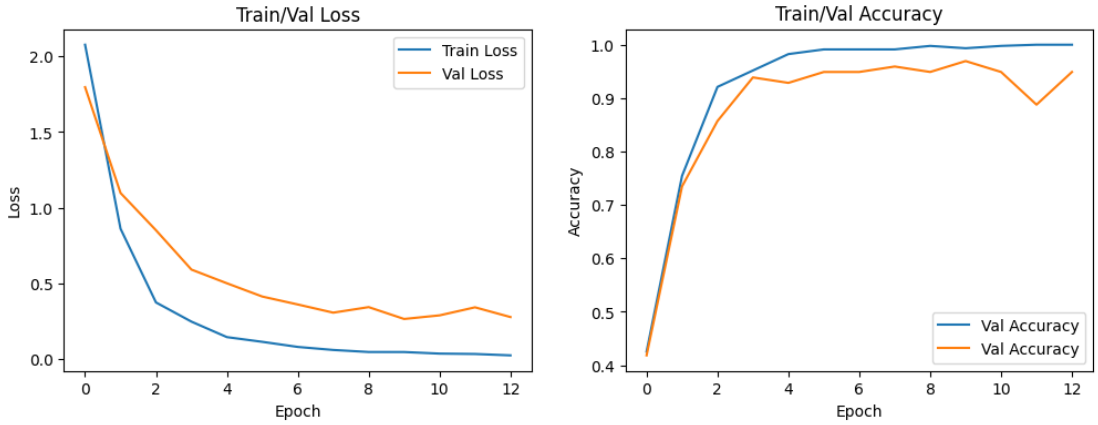
1. **Resize**: All video frames are resized to a fixed spatial resolution (height × width) to ensure consistent input size for the network. The chosen value is **112×112 pixels**, which provides a good balance between computational efficiency and retaining sufficient spatial information for recognizing actions.
2. **Uniform Frame Sampling:** Each video is processed to contain a fixed number of frames. If a video has more frames, a subset is sampled uniformly across the video to preserve temporal coverage. If a video has fewer frames, the last frame is repeated to pad it to the desired length. This ensures that all videos have the same temporal dimension (T) required for 3D CNNs. **64 frames** were chosen as a good balance between capturing temporal dynamics and maintaining computational efficiency, while also matching the pretraining setup of 3D ResNet.
3. **Spatial Augmentation**: For training videos, spatial augmentations are applied to improve generalization. Augmentation introduces variability in the training data, helping the model become more robust to changes in viewpoint, lighting, and appearance. The augmentations are sampled randomly during training, so each iteration may apply a different transformation to the same video, increasing the effective diversity of the training set. Importantly, the same spatial augmentation is applied consistently across all frames within a given video clip, preserving temporal coherence. This design encourages the model to focus on motion and spatiotemporal dynamics rather than frame-specific appearance artifacts or background cues. This includes:
    - **Random Horizontal Flip** (p=0.5): Flips frames horizontally with 50% probability, helping the model handle left-right variations in actions.
    - **Color Jitter:** Randomly adjusts brightness, contrast, saturation, and hue, making the model more robust to different lighting conditions and color variations.
4. **Normalization**: All frames are normalized using the mean and standard deviation of the pretrained 3D ResNet network. Normalization centers and scales the input data, which helps the network converge faster and improves training stability.

4. Model configuration

- **Train test split:** The dataset is divided into training (70%), validation (15%), and test sets (15%). The training set is used to learn the model parameters, the validation set is used for hyperparameter tuning and early stopping, and the test set is reserved for final, unbiased performance evaluation.
- **Weighted Sampling:** A 'WeightedRandomSampler' is used to handle class imbalance, giving rare classes a higher probability of being selected while common classes are sampled less often. This ensures the model is exposed sufficiently to all classes during training and learns to generalize across both frequent and rare categories.

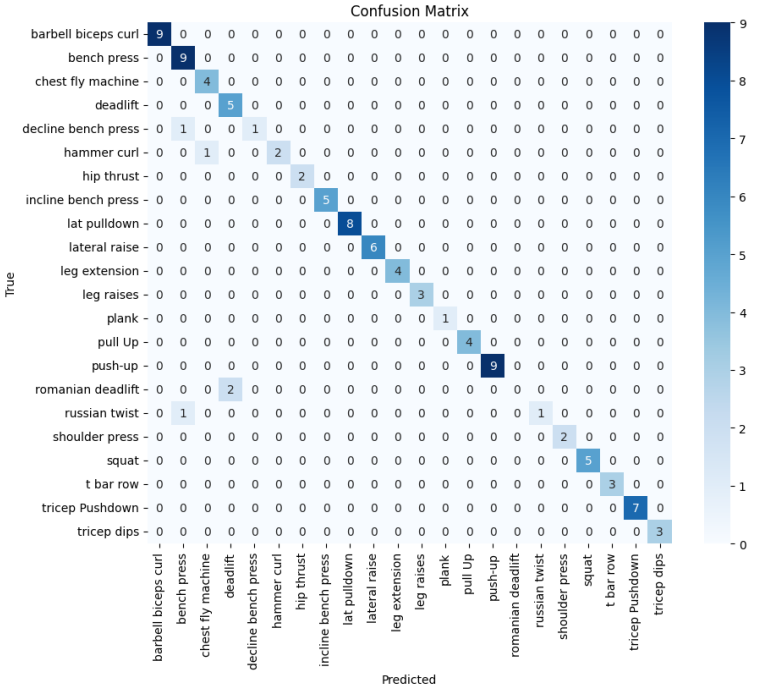
- **Weighted Loss:** Class weights are applied to the loss to compensate for class imbalance, giving more importance to underrepresented classes and helping the model learn them more effectively.
- **Optimizer- Adam:** Adam is used as the optimizer because it combines the benefits of momentum and adaptive learning rates, allowing efficient and stable training even with noisy gradients or small batch sizes. This helps the model converge faster and often leads to better generalization compared to SGD.
- **Early stopping:** Early stopping monitors the validation loss during training and stops if it does not improve by at least 'mindelta' (**1e-2**) for patience (**3 epochs**) consecutive epochs.. This prevents overfitting, a situation where the model learns the training data very well but fails to generalize to unseen examples. To account for the initial unstable phase of training, a minimum number of epochs ('minepochs'=**5**) is required before early stopping begins to be checked, ensuring the model has had sufficient time to start learning meaningful patterns. The saved model corresponds to the epoch with the best validation performance.

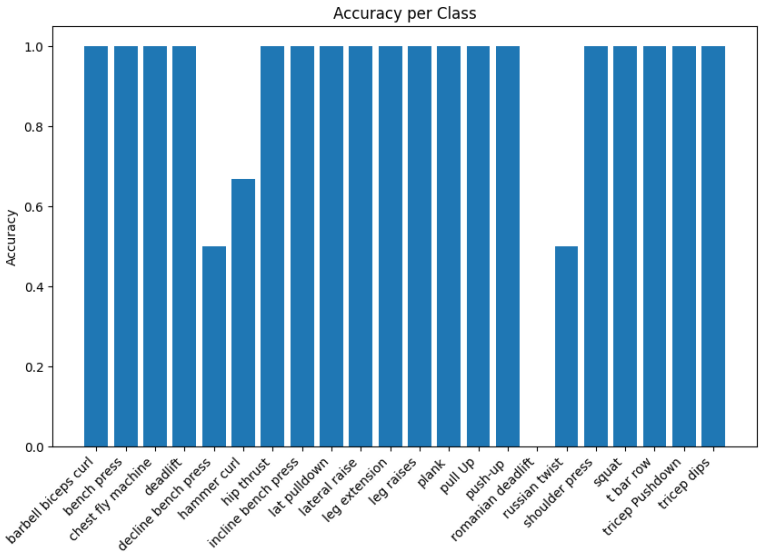
5. Model performance



Training and validation loss and accuracy were monitored throughout training. A gap between training and validation loss is observed, indicating some overfitting. However, the gap remains relatively small, reflecting the effectiveness of the strategies used to mitigate overfitting, such as weighted sampling, early stopping, and fine-tuning only selected layers. Accuracy on both sets improved and stabilized over epochs, showing that the model learned meaningful patterns from the data.

The selected model achieved 99.34% training accuracy, 96.94% validation accuracy, and **94.89% test accuracy** after 10 epochs. The high performance on the test set demonstrates that the model is able to generalize well and has effectively learned to distinguish between the different workout videos, despite the small and imbalanced dataset.





The confusion matrix and per-class accuracy provide deeper insight beyond overall accuracy. Since our dataset is imbalanced (some exercises have very few videos), overall accuracy can be misleading. Showing per-class metrics allows a more precise evaluation of the model's performance on each category.

The confusion matrix shows that most classes are correctly classified, with only a few misclassifications appearing in underrepresented categories. Accuracy per class indicates strong generalization across most exercises, although some classes with limited samples (such as 'russian twist', 'decline bench press', 'romanian deadlift') show lower performance. In contrast, categories like as 'hip thrust', 'leg raises' and 'plank' achieved very high accuracy despite having few examples. This suggests that the methods I used to prevent overfitting in well-represented classes were effective in some cases.

Overall, the results demonstrate that the model effectively learned discriminative spatiotemporal features from the videos, successfully differentiating between a variety of workout types. These findings suggest that selective fine-tuning of higher layers in a pre-trained 3D CNN is sufficient to achieve high performance on a relatively small and imbalanced video dataset.