Baseline:

(ResNet) CNN + MLP = MSE Loss: 10:

A combination of a Convolutional Neural Network (CNN) based on the ResNet architecture and a Multi-Layer Perceptron (MLP). The model is trained using Mean Squared Error (MSE) loss, with a loss value of 10.

ResNet + RNN = MSE Loss: 5:

Another variant of the ResNet architecture combined with a Recurrent Neural Network (RNN). The model is trained with MSE loss, yielding a loss value of 5.

I3D (Inflated 3D ConvNet) + LSTM = MSE Loss: 4:

I3D (Inflated 3D ConvNet) is a 3D extension of the ConvNet, along with a Long Short-Term Memory (LSTM) network. The model is trained with MSE loss and achieves a loss value of 4.

Model:

1. The model begins by breaking down the input video into separate visual and audio tracks.

2. **Unsupervised Pretraining - Visual Track**: The visual track of the video is processed using a ConvLSTM Autoencoder. This architecture combines convolutional layers with LSTM (Long Short-Term Memory) layers to capture both spatial and temporal dependencies in the visual data. The autoencoder is trained in an unsupervised manner, meaning it learns to reconstruct the input data without any specific labels. The output of this process is an encoded representation or embedding of the visual context of the video.

**Unsupervised Pretraining - Audio Track**: The audio track of the video is processed using a pretrained model called DeepSpeech, developed by Mozilla or the open-source Whisper. The model converts the audio into a corresponding textual transcript, providing additional information about the content of the video. This step is prebuilt, meaning it uses a pre-existing model rather than training from scratch.

3. **Embedding Generation - Visual Embedding**: The encoded visual representation from the ConvLSTM Autoencoder is extracted as an embedding vector, capturing the visual context of the video.

**Embedding Generation - Audio Embedding**: The DeepSpeech/Whisper model generates textual output from the audio track, representing the spoken content. This output can be considered as an embedding of the audio information.

4. **Concatenation and Feature Fusion**: The visual and audio embeddings are concatenated or combined into a single feature representation and fed into a transformer-based regression model. This fusion of modalities allows the model to leverage both visual and audio information for subsequent analysis. The transformer consists of LSTM or MLP layers.:

5. **Transformer-Based Regression**: The concatenated feature representation is input into a transformer-based regression model. The transformer architecture, commonly used in natural language processing tasks, is adapted here to handle the combined visual and audio data. It learns the semantic and non-linear relationships between the input features and the video creator success metrics, such as video views. The regression model is trained to predict these metrics based on the fused feature representation.

6. **Loss Calculation**: The model is trained using a mean squared error (MSE) loss function, with the goal of minimizing the discrepancy between the predicted values and the actual values. The final MSE loss target is 0.01, indicating a high level of accuracy in predicting the video creator success metrics. The output of the regression model is compared to the ground truth video creator success metrics using the mean squared error (MSE) loss function. The aim is to minimize the difference between the predicted values and the actual values.

7. **Baseline and Evaluation**: To evaluate the effectiveness of the proposed model, a baseline model is established using a simpler architecture or approach. The baseline is compared with the main implementation in terms of the achieved MSE loss. A lower MSE loss value indicates better performance in predicting video creator success metrics.

Overall, this model leverages unsupervised pretraining, pretrained audio models, and a transformer-based regression model to predict video creator success metrics, with a focus on video views. The combination of visual and audio information allows for a more comprehensive understanding of the video content and improves the accuracy of the predictions. The model architecture includes elements of both unsupervised pretraining and supervised fine-tuning, combining the benefits of both approaches. The model also incorporates a process called dual transfer learning, where both the visual and audio embeddings contribute to the final prediction.

Experiment:

Divide the dataset into three parts: training set (70%), validation set (10%), and testing set (20%).

Experiment with different multimodal fusion techniques, such as **early fusion** (combining features before feeding them to the regression model) and **late fusion** (independently training individual models for each modality and then combining their predictions).

**Evaluate the performance** of the model on the testing set using the chosen evaluation metrics (e.g., MSE). Compare the results of different fusion methods to identify the most effective approach.

Conduct **ablation experiments** to study the **impact of each modality** on the prediction performance. Train the model using only video features, only audio features, only textual metadata, and then compare the results to the multimodal fusion approach.

Perform **hyperparameter tuning** to find the **optimal** architecture and parameters for the regression model.

Analyze the model's predictions to gain insights into the **key factors** influencing TikTok content success. **Visualize the attention mechanisms** to understand which parts of the video, audio, or textual information are most relevant for popularity prediction.

Hyperparameter tuning is an essential step in the process of training machine learning models to achieve better performance on a specific task. In the context of the "Multimodal Deep Regression on TikTok Content Success" task, hyperparameter tuning involves finding the optimal values for various hyperparameters that govern the behavior of the deep regression model. The goal is to improve the model's ability to predict the success or popularity of TikTok content accurately.

Here are some key hyperparameters that could be tuned for the multimodal deep regression model:

**Learning Rate**: The learning rate determines the step size at which the model updates its parameters during training. A small learning rate may lead to slow convergence, while a large learning rate may cause the model to overshoot the optimal solution. Commonly used optimization algorithms like Adam and SGD have learning rate parameters that need tuning.

Number of Hidden Layers and Units: The architecture of the deep regression model can greatly impact its performance. This includes determining the number of hidden layers and the number of units (neurons) in each layer. Too few layers or units may result in underfitting, while too many may lead to overfitting.

**Dropout Rate**: Dropout is a regularization technique used to prevent overfitting. It randomly deactivates neurons during training, forcing the model to rely on different pathways for making predictions. The dropout rate determines the probability that a neuron is deactivated.

**Batch Size** (1): The batch size is the number of samples used in each iteration of the training process. A larger batch size can lead to faster convergence but may require more memory, while a smaller batch size can increase training time but may lead to better generalization.

**Number of Epochs**: The number of epochs defines how many times the model will iterate over the entire dataset during training. Too few epochs may result in underfitting, while too many may lead to overfitting.

**Regularization Strength** (Transformer regularization L1 and L2

): L1 and L2 regularization can be applied to the model to penalize large weights. The regularization strength determines the impact of the regularization term on the loss function.

Activation Functions: The choice of activation functions in the model's hidden layers can impact its ability to capture complex patterns in the data. Common choices include ReLU, tanh, and sigmoid functions.

Image and Video Preprocessing: The preprocessing steps for images and videos may include resizing, cropping, normalization, and data augmentation. These steps can also have an impact on the model's performance.

**Define the Search Space**: Before starting the hyperparameter tuning, it is essential to define the search space for each hyperparameter. For example, the learning rate could be searched in a range from 0.0001 to 0.1, the batch size could be chosen from [32, 64, 128], and the number of layers could be in the range [1, 3, 5]. Each hyperparameter should have a predefined set of possible values or distributions.

To tune these hyperparameters, there are several techniques.

* **Grid Search**: It exhaustively searches all possible combinations of hyperparameters from the predefined search space. While it's straightforward, it could be computationally expensive and time-consuming.
* **Random Search**: Randomly samples hyperparameters from the predefined search space. It is less computationally intensive compared to grid search and often yields similar results.
* **Bayesian Optimization**: Bayesian optimization builds a probabilistic model of the objective function and chooses the next hyperparameter combination based on the previous observations. It is often more efficient in finding optimal hyperparameters compared to grid or random search.
* **Genetic Algorithms**: Genetic algorithms use techniques inspired by natural selection to evolve a population of hyperparameter combinations over generations.

**Early Stopping**: Early stopping is a technique used during training to prevent overfitting. It stops training the model when the performance on the validation set starts to degrade.

During hyperparameter tuning, it is crucial to use proper validation techniques like cross-validation to avoid overfitting to the validation set. The process may involve multiple iterations of training and validation until the best combination of hyperparameters is found. Once the optimal hyperparameters are determined, the model can be trained on the entire dataset using these values to make predictions on new TikTok content and estimate their success.

**Regularized regression** is a variant of linear regression that introduces limitations or corrections to the coefficient estimates to avoid them from approaching zero. Essentially, regularization helps reduce the computational burden during model operation and mitigates non-uniform stability issues that may arise during the learning process. By doing so, it effectively eliminates the risk of overfitting. The regularization function incorporates additional terms into the original loss function, effectively acting as a penalty for the complexity of the model.

In this context, L2 regularization is specifically employed to combat overfitting and enhance the generalization capability of the model. It ensures that the model's parameters do not become overfitted and helps reduce the risk of overfitting. By adjusting the parameter 𝛼 in the L2 regularization term, the weight matrix of the model is controlled. A higher 𝛼 value results in smaller weights for many hidden elements in the model, making their impact negligible. This process essentially reduces the neural network to a smaller version while maintaining the network's depth. As a result, the model transitions from an overfitting state to an underfitting state.

In the context of human action recognition, the L2 regular joint attention mechanism is employed. L2 weight decay regularization is a common technique used in deep learning, particularly to address overfitting issues caused by an excessive number of features. The solution involves reducing the weight of features or penalizing unimportant features. Regularization can effectively prevent overfitting and enhance the model's ability to generalize.

When training the TimeSformer model, applying L2 weight decay regularization can prevent overfitting on the training data and improve the model's generalization capabilities. Specifically, L2 regularization introduces the L2 norm into the loss function, which penalizes parameters and achieves the desired regularization effect.

A diagram and a diagram of a function

Description automatically generated