Master in
Computer Vision
Barcelona

Multimodal Human Analysis

Module: C5

Group: 7

Students: Cristian Gutiérrez
Iñaki Lacunza
Marco Cordón
Merlès Subirà

Index

- Task a) **Different sets statistics**
- Task b) **Define a classifier with image only and evaluate**
- Task c) **Define a train strategy**
- Task d) **Define a test strategy**
- Task e) **Define a strategy to represent acoustic data**
- Task f) **Define a strategy to represent text data**
- Task g) **Multimoda model**
- Task h) **Ablation study**
- **Summary Slide**

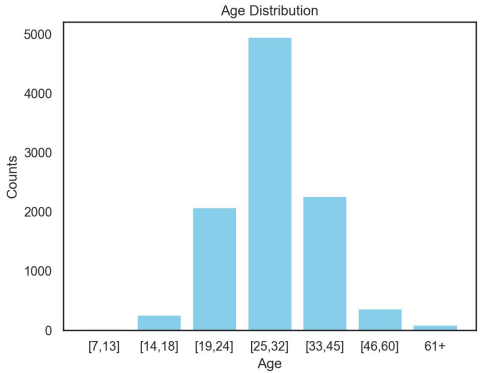
Task a. Different sets statistics

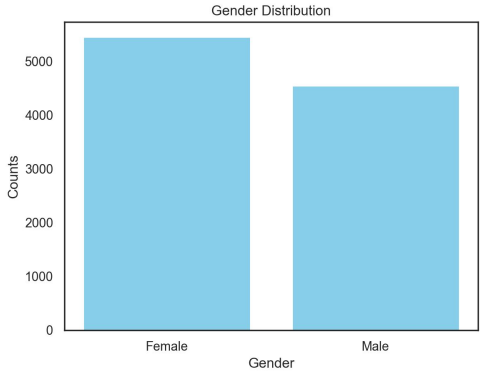
Task a. Different sets statistics.

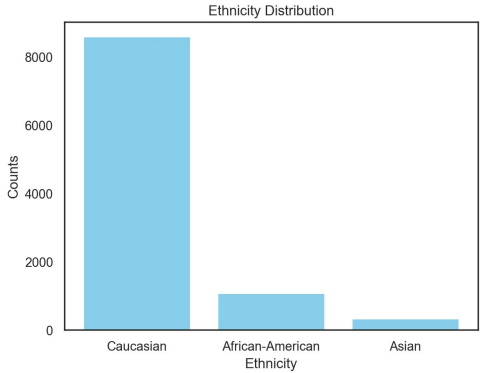
First thing we did was to create our own `DataLoader` class and take into account the full dataset without partitions to compute the Histograms for each label AgeGroup, Gender and Ethnicity.

```
full_dataset = ConcatDataset([train_partition, val_partition, test_partition])
```

Then we first computed the frequencies of each individual objective variable:





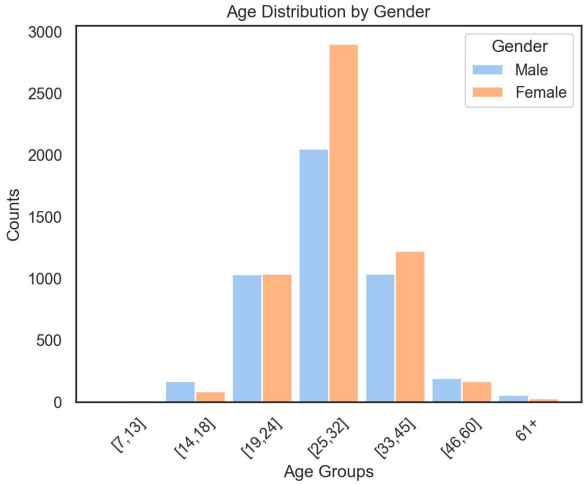


The age distribution seems to follow a normal distribution, by having the majority of the data centered around the central age range [25, 32]. Also, there is a huge predominance on Caucasian in the Ethnicity.

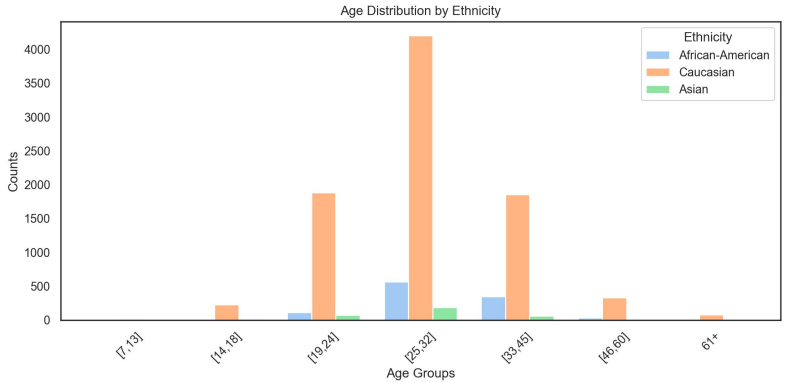
On both `train`, `val` and `test` partitions this claims we've just made for the fully hold.

Task a. Different sets statistics.

We then computed the **age distribution** by our other two objective variables ‘Gender’ and ‘Ethnicity’.

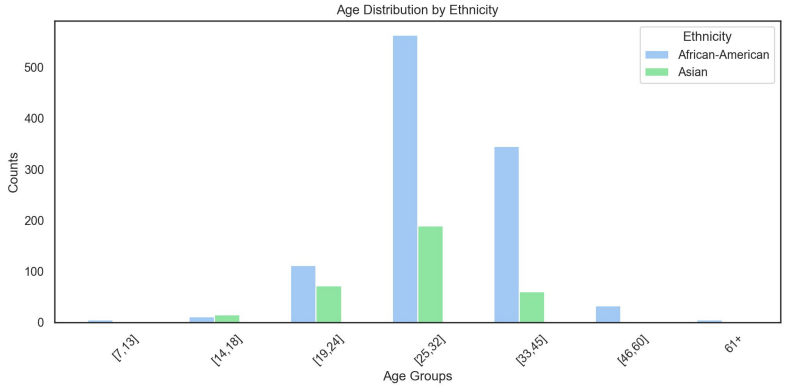


Both genders follow a similar **normal** –like distribution. Also, they are more or less balanced thus we do not draw any problems.



Both genders follow a similar **normal** –like distribution. Also, they are more or less balanced thus we do not draw any problems.

Age Distribution without Caucasian



For clarity we plot the age distribution for the Ethnicity case without Caucasian and we can see the **imbalance** in age for African-American subjects.

Task a. Different sets statistics.

We indicate in underscore maximum of each age range, and we add the difference of each age with respect to the baseline (the ethnicity that has the maximum frequency) per each range.

We also mark in yellow those cases where there is NO representation.

Ethn.	7–13	14–18	19–24	25–32	33–45	46–60	61+
Caucasian	14 (+0%)	228 (+0%)	1885 (+0%)	4201 (+0%)	1854 (+0%)	328 (+0%)	80 (+0%)
African-A.	5 (-64%)	11 (-95%)	112 (-94%)	563 (-87%)	345 (-81%)	33 (-90%)	5 (-94%)
Asian	0 (-100%)	15 (-93%)	72 (-96%)	189 (-96%)	60 (-97%)	0 (-100%)	0 (-100%)

We can see how we don't have **any representation** for **Asian** subjects of ages **[7–13], [46–60] and 61+**.

Furthermore by the observed differences we can see dangerous levels for the same age ranges of African-Americans, and for both ethnicities on [14-18] years.

Task a. Different sets statistics.

Gender / Ethn.		7–13	14–18	19–24	25–32	33–45	46–60	61+
F	Caucasian	8 (+0%)	72 (+0%)	919 (+0%)	2378 (+0%)	929 (+0%)	149 (+0%)	29 (+0%)
	African-A.	4 (-50%)	7 (-90%)	69 (-92%)	390 (-84%)	244 (-74%)	19 (-87%)	0 (-100%)
	Asian	0 (-100%)	6 (-92%)	49 (-95%)	134 (-94%)	50 (-95%)	0 (-100%)	0 (-100%)
M	Caucasian	6 (+0%)	156 (+0%)	966 (+0%)	1823 (+0%)	925 (+0%)	179 (+0%)	51 (+0%)
	African-A.	1 (-83%)	4 (-97%)	43 (-96%)	173 (-91%)	101 (-89%)	14 (-92%)	5 (-90%)
	Asian	0 (-100%)	9 (-94%)	23 (-98%)	55 (-97%)	10 (-99%)	0 (-100%)	0 (-100%)

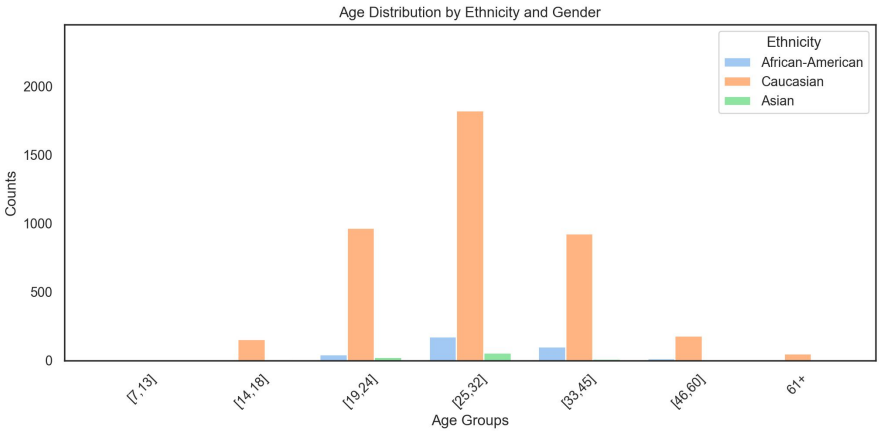
We can see how **African-A** is under-represented, and within them **Male** is worse represented than **Female**.

Asian is the worst ethnicity in terms of representation, and **Male** is worse represented than **Female** also.

Task a. Different sets statistics.

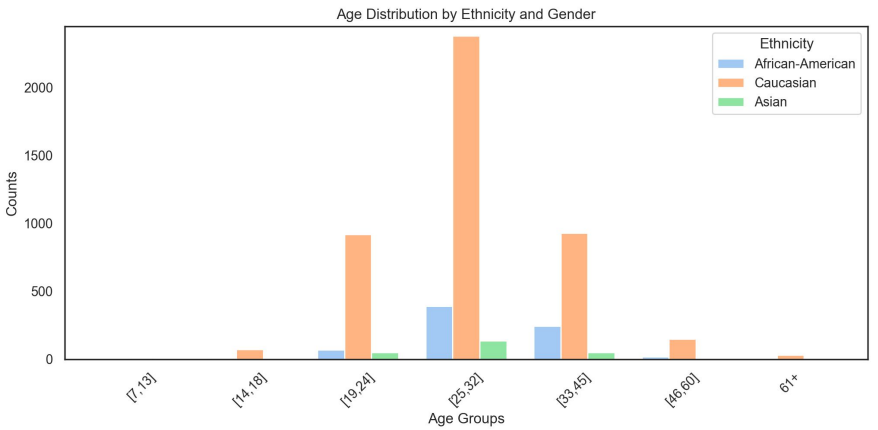
We show both age distribution for gender and ethnicity with 2 plots sharing the same scale:

Age Distr. by Ethnicity for Gender = Male



We can see how the age distribution is very similar by ethnicities, being the caucasian the ethnicity which more users by a great difference. Moreover, asian male users there are only a few between 19 and 32 years.

Age Distr. by Ethnicity for Gender = Fem.



The female distribution is very similar to the male one but with a bigger number of users. Now, there are images of asian users in the range of 33-45 years.

Task b. Train a classifier with image only and evaluate

Task b. Image only classifier

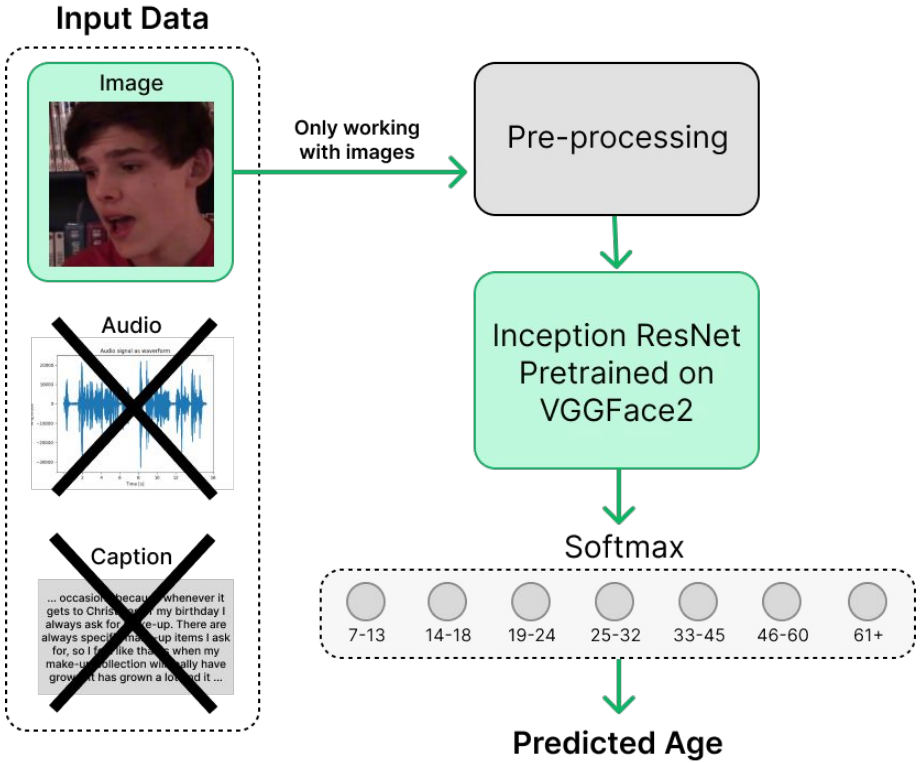
MODEL OVERVIEW

We have used the `baseline Inception Resnet V1` model given to us to do this task. The model uses pre-trained weights for the large-scale face recognition VGGFace2 dataset containing +3.3 million samples.

As pre-processing, all images are resized to a common size of 224, then several data augmentations `TrivialAugmentWide` and a normalization step for each of the images.

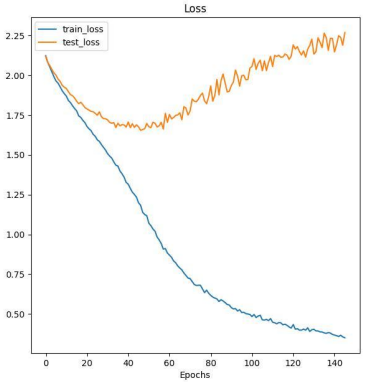
For training, the images are batched with a size of 256, and 300 epoch with an early stopping of 50.

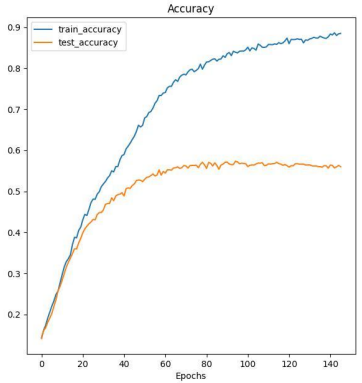
For the loss, Cross-Entropy func is used to compute the logits, which are then processed through a softmax function to yield the final age predictions.



Task b. Image only classifier

If we monitor the loss obtained for both stages of *training* and *testing* we can see how for the loss, the model is able to converge for the training partition, however for the test set, the loss fluctuates (which is somewhat normal) but overall it increases as the training loss decreases. This is a first indicative sign of **overfitting**!





On the other hand if we monitor the accuracy, we see that the training accuracy keeps increasing in a logarithmic manner which is a sign of healthy training, and the testing accuracy performs the same way but leading to a **gap** between both of them. Testing stays at 0.55~ meanwhile training keeps on going on the 0.9~ plateau.

Furthermore, we can see how the early stopping policy is monitoring the testing accuracy.

Overall, both the loss and the accuracies look healthy. There is overfitting being suffered possibly memorizing some training instances that are not being translated onto testing knowledge.

Task b. Image only classifier

Having a batch size of 256 is quite big, and it is generally recommended to always use the biggest batch size our GPU accepts as a rule of thumb. But we wanted to check how much this hyperparameters affects and we re-trained the model just by changing the batch size to 32.

What if we use a smaller batch size of 32?

DEFAULT BATCH SIZE OF 256

- Epoch until stop: **145**
- Avg Test Accuracy: **0.5559**

TRAINING TIME: 1h 13min

✅

BATCH SIZE OF 32

- Epoch until stop: **77**
- Avg Test Accuracy: **0.2906**

TRAINING TIME: 37min

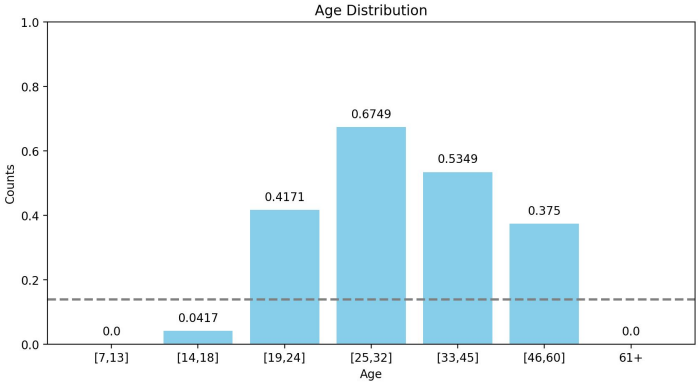
❌

Thus, even though our early stop criteria triggers much faster in advance with a low batch_size, the model has not been able to generalize at all, at the cost of having a big leap in accuracy.

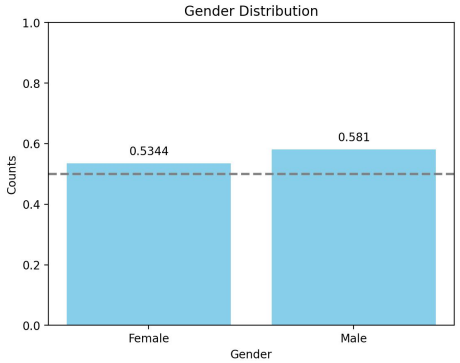
Task b. Image only classifier

In terms of evaluation, we used the `evaluate.py` script to assess the performance of the model. This script calculates the global average accuracy as well as category-specific accuracies based on age, gender, and ethnicity. *(We indicate with a dash line better than chance)*

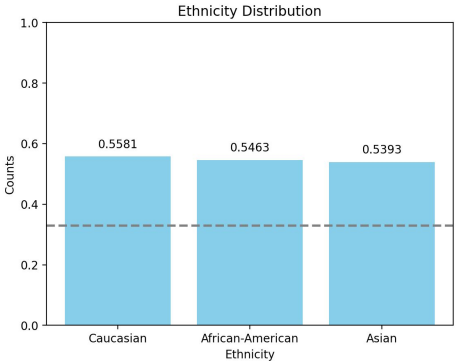
AGE CLASSIFICATION ACCURACIES



ACC BY GENDER



ACC BY ETHNICITIES



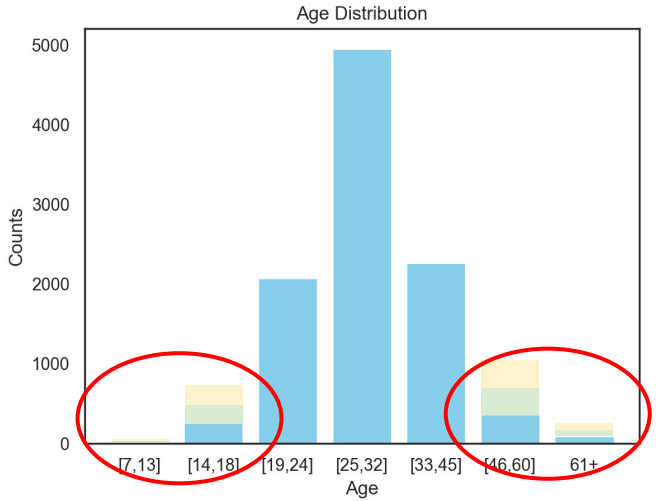
Also, we obtain a mean age bias of **0.12** which might indicate some discrepancy on the fairness of the classifications results, more precisely we obtain 0.3305 of bias for Caucasian, which is explained by the vast difference in frequencies of samples w.r.t. other ethnicities.

Task c. Define a train strategy.

Task c. Define a train strategy.

Analyzing the task_a results, we have confirmed the **big unbalance that our dataset has**. In order to try to reduce this unbalance **without reduce the number of train instances** that we are using in the training, we have proposed **2 different strategies**:

- **Data augmentation.**
- **Custom loss.**



Data Augmentation:

In the data augmentation strategy we **only create new images for the classes that have a very low number of training images** compared with the other classes. These classes are: **1, 2, 6, 7**.

We employ the v2.transformation from PyTorch to create new images based on the ones we already have. We create 2 sets of new train images:

- The original images but with **Horizontal Flip**.
- Images with **Random Horizontal Flip + Random Rotation(-12º,12º) + Small change of brightness.**

Number of images in old train set: **6011**

Number of images in new train set: 6011+ 457+ 457 = **6925**

Task c. Define a train strategy.

First new test set: (457 images)

```
augment_transform_1 = transforms.Compose([
    v2.Resize(size=IMAGE_SIZE),
    # METER TRANSFORMS,
    v2.TrivialAugmentWide(),
    v2.RandomHorizontalFlip(p=1),
    v2.ToTensor(),
    v2.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

Original
Size: (224, 224)

Transformed
Size: torch.Size([224, 224, 3])

As we can see, in this new set of images we only apply an horizontal flip. The change of color in the transformed image is due to the representation after the normalization step, but there is not any color change in this group of images. **Our main goal is to increase the smallest train sets but trying to avoid the overfitting**.

Second new test set: (457 images)

```
augment_transform_2 = transforms.Compose([
    v2.Resize(size=IMAGE_SIZE),
    # METER TRANSFORMS,
    v2.TrivialAugmentWide(),
    v2.RandomHorizontalFlip(p=0.5),
    v2.RandomRotation(degrees=(-12, 12)),
    v2.ColorJitter(brightness=.1, hue=.005),
    v2.ToTensor(),
    v2.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

Original
Size: (224, 224)

Transformed
Size: torch.Size([224, 224, 3])

In this second set we applied more transformation like a rotation or a color change. The **color change is very limited** because we do **not want to confuse the model in the ethnicity parameter.** The rotation is important since not always the faces are straight in the image.

Task c. Define a train strategy.

The problem of the **Data-augmentation** strategy is that **is very limited** due to the **big difference between the amount of train images between classes** (class 1: 19 images ↔ class 4: 2932 images). But on the other hand, we think that i**s important to use all the resources** that we have to train the model since the test set has the same distribution. In an attempt of **improving more our training** we have applied a **custom loss**:

Custom Loss:

To avoid a bit more the unbalance of the training data, we use a **weighted loss function**. We pass to the loss function a weight vector in which each class has assigned one value that is inversely proportional to the number of train images of that class. In our case the vector is:

[30, 492, 1264, 2932, 1353, 696, 153] ➗ 6925

Accuracy Results:

After the implementation of the two strategies and the combination of both, we have obtained the next results:

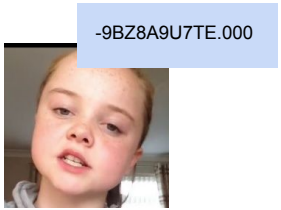
Baseline Train Strategy (Test Acc): **0.52**
New Train Strategy (Test Acc): **0.54**

As we can see **the results have improved a bit** with this new implementation, so it would be nice to include in the next models.

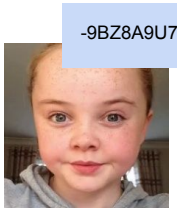
Task d. Define a test strategy.

Task d. Define a test strategy.

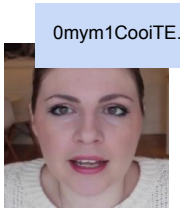
Once the train strategy is ready, it is time to define a test strategy to evaluate our model. In the **baseline model implementation**, the strategy is to **treat all the images of each user individually** as if they were not related to each other. And at the end aggregate all the test images predicted results to calculate the **Test Accuracy**:



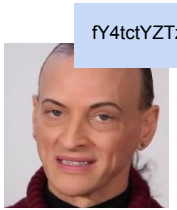
Pred Label: 3
Gt Label: 1



Pred Label: 1
Gt Label: 1



Pred Label: 3
Gt Label: 3



Pred Label: 4
Gt Label: 7

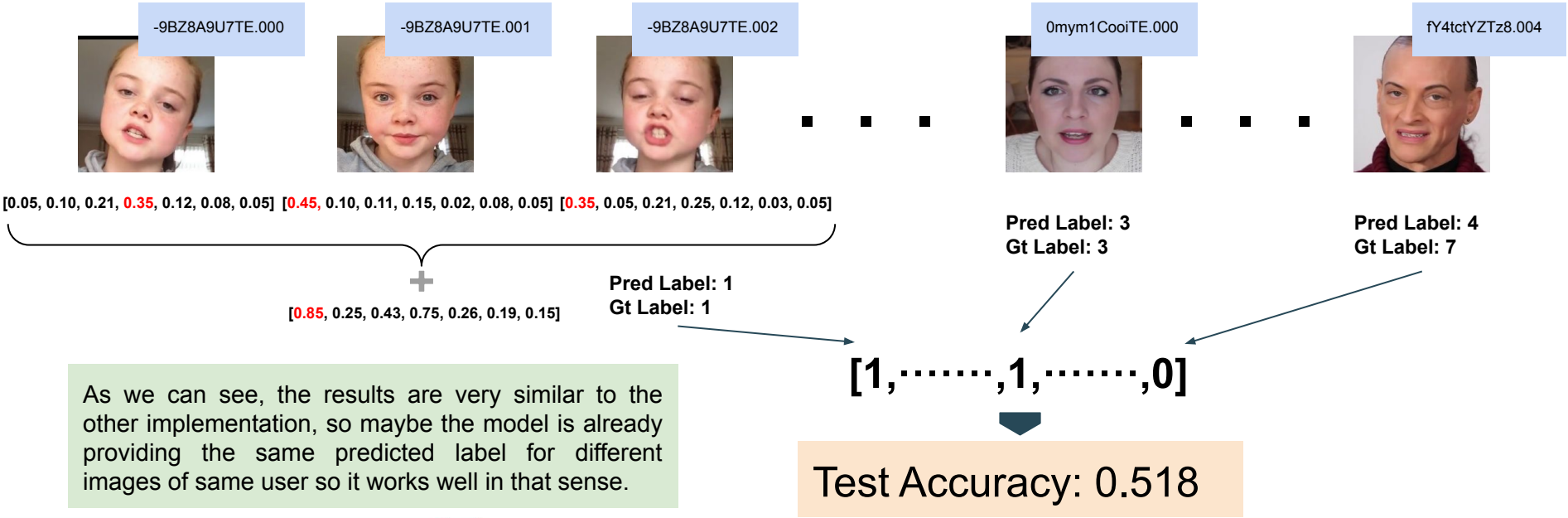
[0,1,.....,1,.....,0]

One-hot vector:
True=1
False=0

Test Accuracy: 0.523

Task d. Define a test strategy.

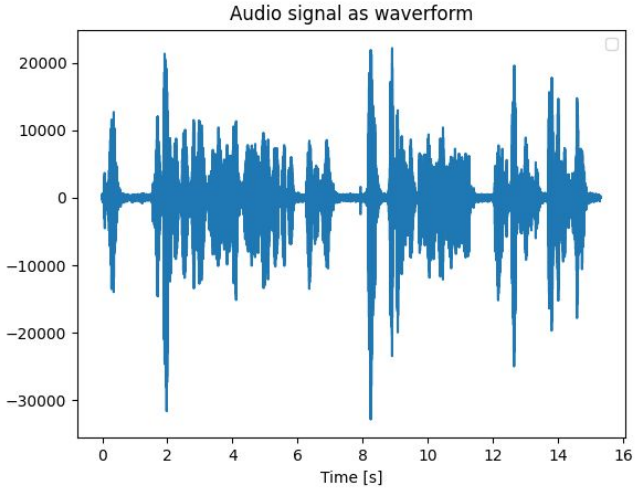
We have created **another test strategy**. In our case we think that **if we aggregate the results of the same users images we can avoid some false predictions**. To do this, we add the probability lists of all the images with the same User ID, and then we use argMax to keep the label that has the most probability in the aggregate.



As we can see, the results are very similar to the other implementation, so maybe the model is already providing the same predicted label for different images of same user so it works well in that sense.

Task e.
Define a strategy to represent acoustic data

Task e. Define a strategy to represent acoustic data.



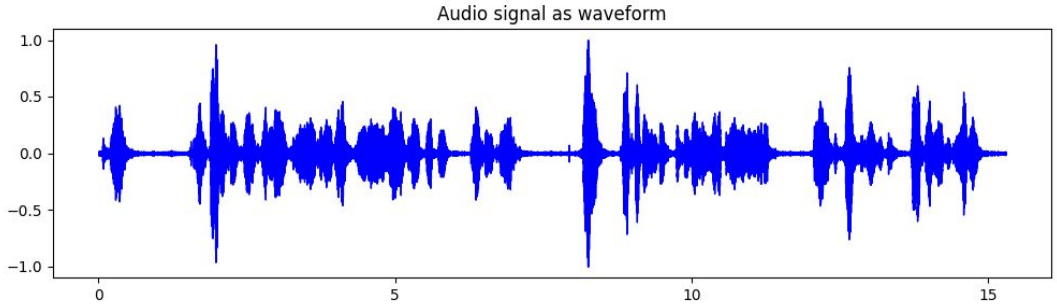
The first step consisted on representing the waveform of the audio signals in order to get a better understanding of the problem we had to work on.

Afterwards, in order to perform feature extraction with the audio data, we decided to work with the Librosa library.

The data extracted using the *Librosa* is computed by dividing the original values by 2^15-1 (because we have 16bit audios), which is a better way of working with the data, since the saved volume may be different depending on the used device.

Plot computed using ***'wavfile'*** (code provided by the teacher)

Plot computed using ***'Librosa'***



Task e. Define a strategy to represent acoustic data.

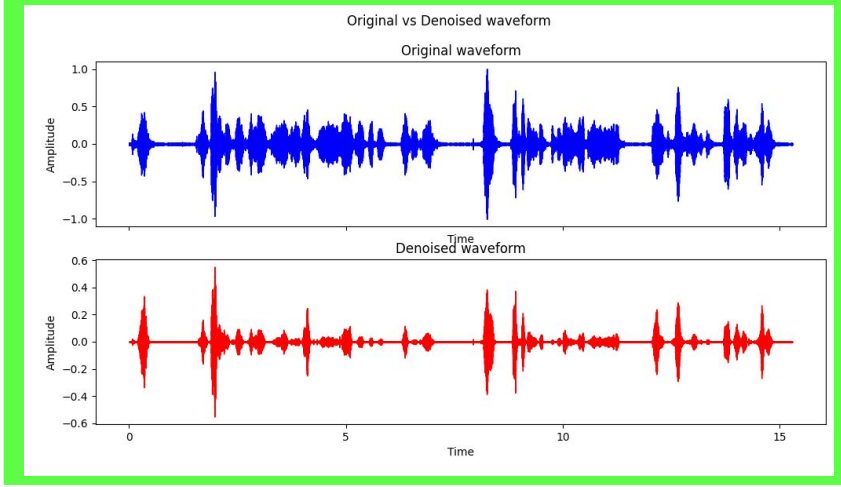
The duration is fixed for all the audios (15.302s), and we decided to fix the ***samplerate*** in 8000.

There is lots of *characteristic* information stored in the waveform:

- Number of words in the audio.
- Number of words per second.
- Tempo
- Fundamental frequency:
  - Mean
  - Median
  - Standard deviation
  - 5-percentile
  - 95-percentile

But before starting extracting the features, some preprocessing steps are necessary:

DENOISING: Performed using the Noisereduce library.



TRIMMING:

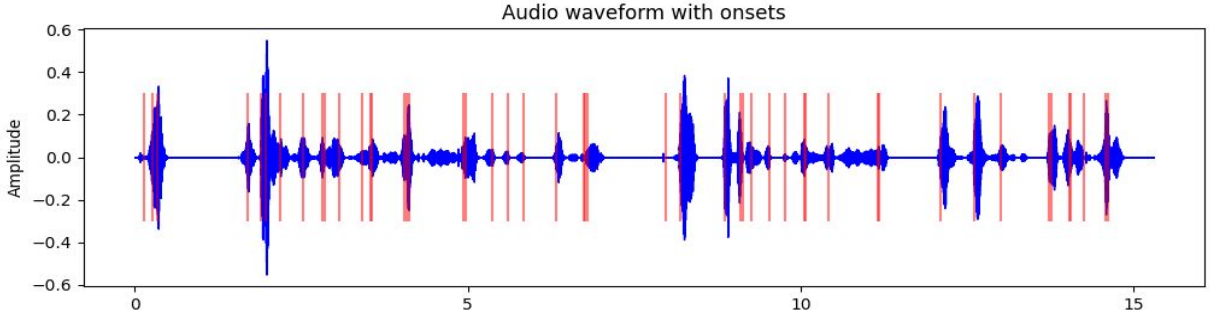
There also exists the possibility of cutting the audio at the beginning and at the end, getting rid of very low values. This could easily be made using the following line:

```
trimmed_data, _ = librosa.effects.trim(denoised_data, top_db=20)
```

But, we decided not to implement this preprocessing step, **because we consider that the pause or the urge before starting to talk is a characteristic of the age of the speaker.** For instance, old people talk more paused and may need more time to start talking, and, on the other hand, children tend to start talking very fast.

Task e. Define a strategy to represent acoustic data.

By finding the ***onset*** locations (beginning of of a sound) it is possible to calculate the number of words in the audio, and hence to compute the ***number of spoken words per second.***
On the other hand, the ***tempo*** of the audio can be extracted using `tempo = librosa.beat.tempo(y=data_noise_reduced, sr=SAMPLERATE, start_bpm=10)[0]`
Obviously, with some exceptions , but we expect younger people to have higher values of ***number of words per second*** and ***beats per minute.***

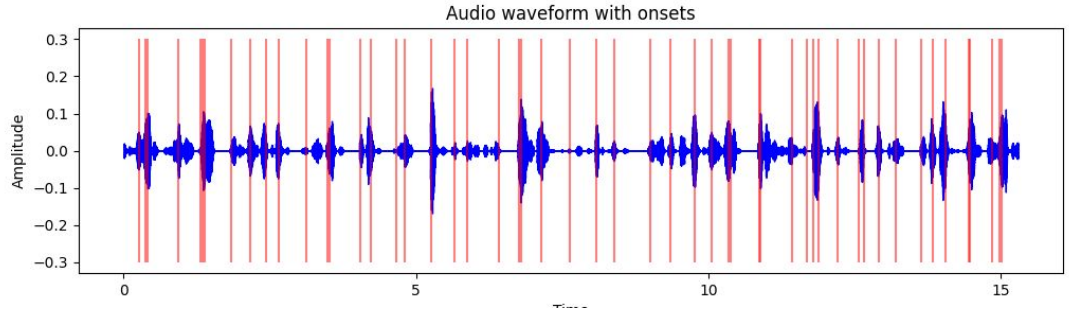


Age class: 6 → 46-60 years

Audio length: 15.302s
Number of words/s: 3.20
Beats per minute (bpm): 18.38

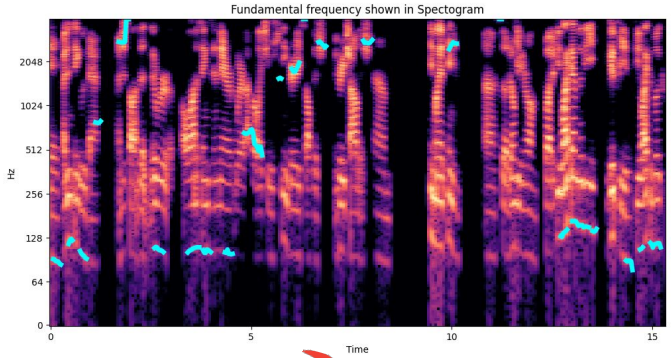
Age class: 2 → 14-18 years

Audio length: 15.302s
Number of words/s: 3.46
Beats per minute (bpm): 21.31



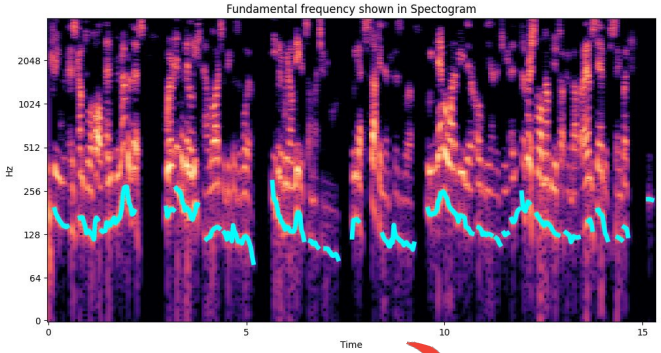
Task e. Define a strategy to represent acoustic data.

Fundamental frequency: Lowest frequency at which a periodic sound appears (known as pitch in music). In the spectogram plots, the fundamental frequency is the lowest horizontal strip, and the repetition of the stip pattern above this fundamental are called harmonics.



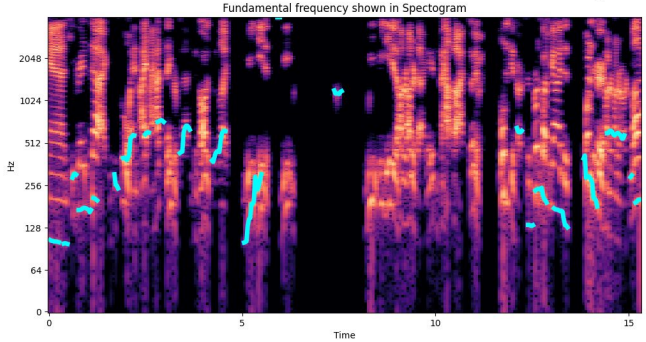
Age class: 1 → 7-13 years

f0 mean:	443.26
f0 median:	292.60
f0 standard deviation:	544.91
f0 5-percentile:	102.56
f0 95-percentile:	1145.30



Age class: 4 → 25-32 years

f0 mean:	971.51
f0 median:	151.46
f0 standard deviation:	1242.97
f0 5-percentile:	89.64
f0 95-percentile:	3993.93



Age class: 7 → 61+ years

f0 mean:	154.15
f0 median:	145.46
f0 standard deviation:	39.86
f0 5-percentile:	104.68
f0 95-percentile:	233.51

Task e. Define a strategy to represent acoustic data.

Audio pipeline scheme:
Extract wanted features (task e) & use them as input for a age classification model (task g)

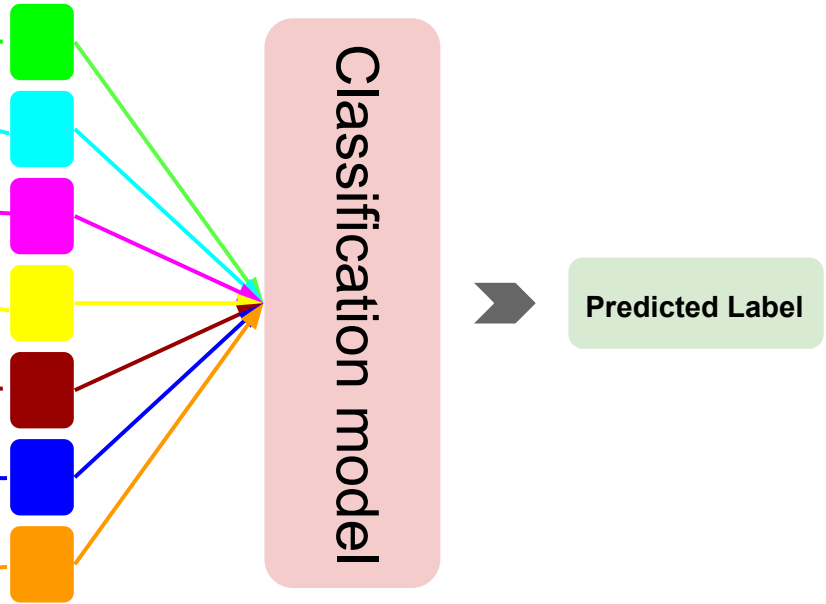
The number of words in the audio depends on the sentence itself, **it is a characteristic of the text, not from the speaker**. So we will not use it as input for our classification model.

1st step: Preprocess data:

- Divide data by 2^15
- Noise reduction

2nd step: Extract features:

- # words in the audio ✗
- # words per second
- # beats per minute
- Fundamental frequency:
  - Mean
  - Median
  - Standard Deviation
  - 5-percentile
  - 95-percentile

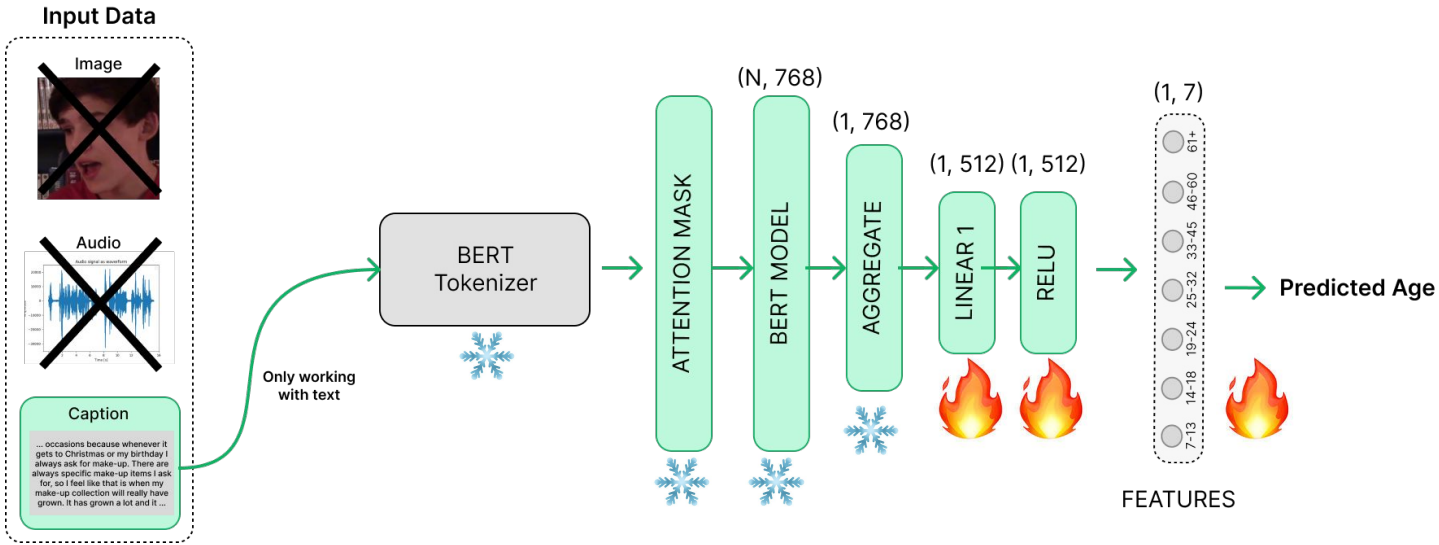


Task f. Define a strategy to represent text data.

Task f. Define a strategy to represent text data.

As a textual encoder to extract features from text we've used BERT from the HuggingFace transformer's library that gives us pre-trained core models and tokenizers already working fine.

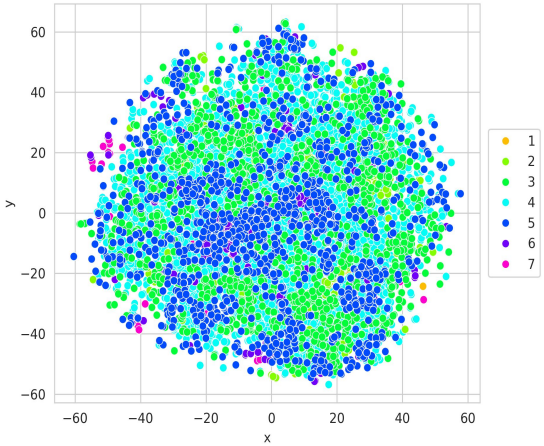
We will use BERT as the encoder and add 2 non-linear layers at the top with ReLU activations to break linearity and try to establish some nice representation learning for the captions.



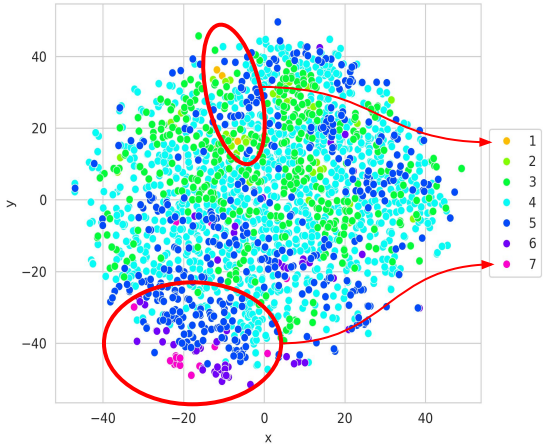
We will only actively train the final layers `LINEAR 1`, `ReLU` and `LINEAR 2`!!!

Task f. Define a strategy to represent text data.

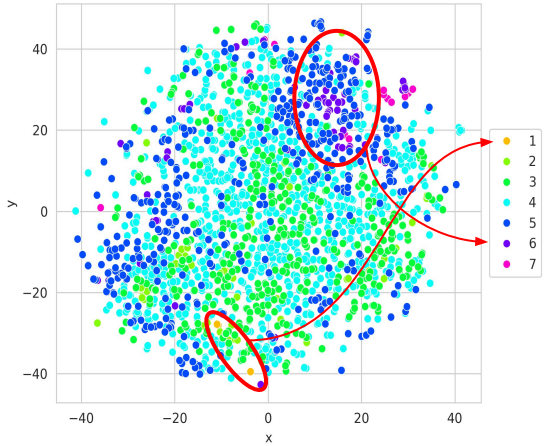
Train TSNE



Val TSNE



Test TSNE

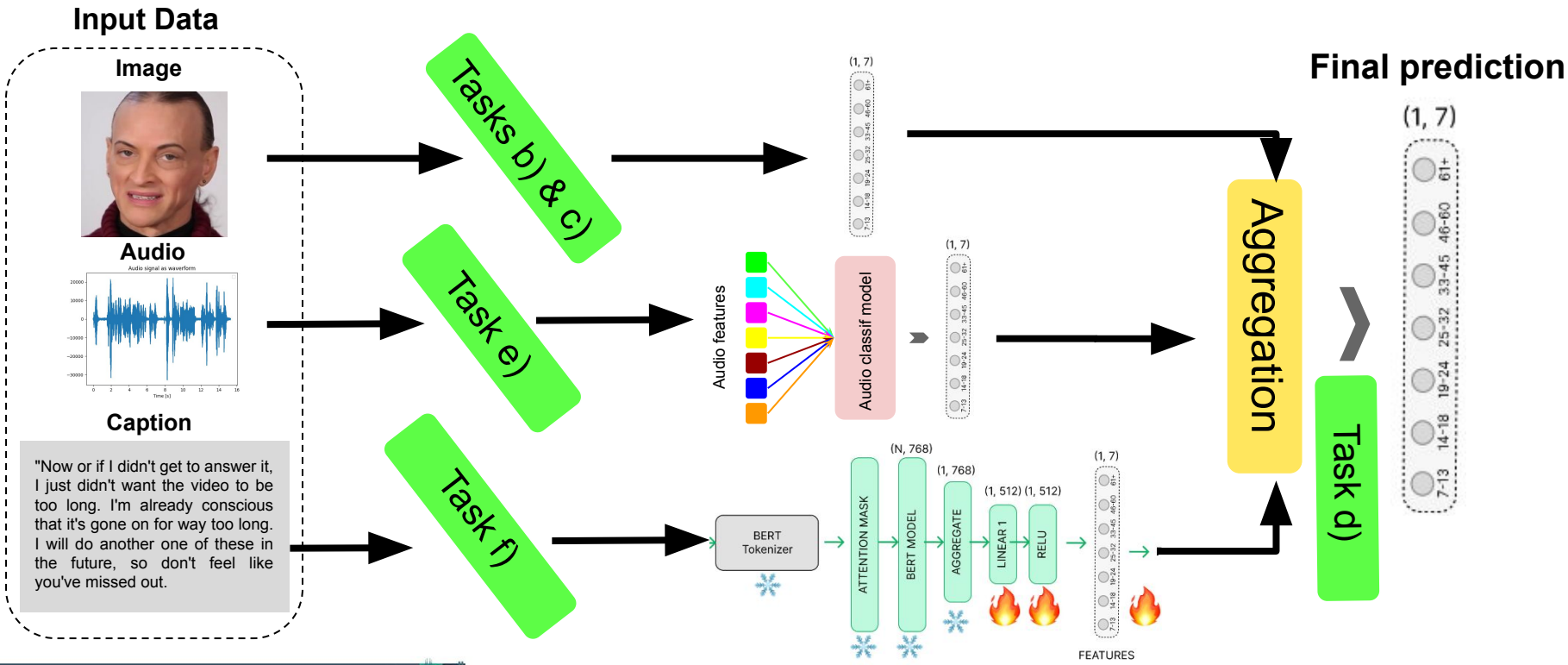


At, first sight, we do not expect to get a good classifier using just text, since text information does not generally offer much information about the age of the writer. As it can be seen, all classes are very spreaded out.
But, for some very specific age intervals, such as old or very young people (classes 1→ [7, 13], 6 → [46, 60], 7 → 61+), their embeddings are generally more concentrated in specific locations.
So, **we foresee that text information will be important for just some specific age intervals.**

Task g. Multimodal model.

Task g. Multimodal model.

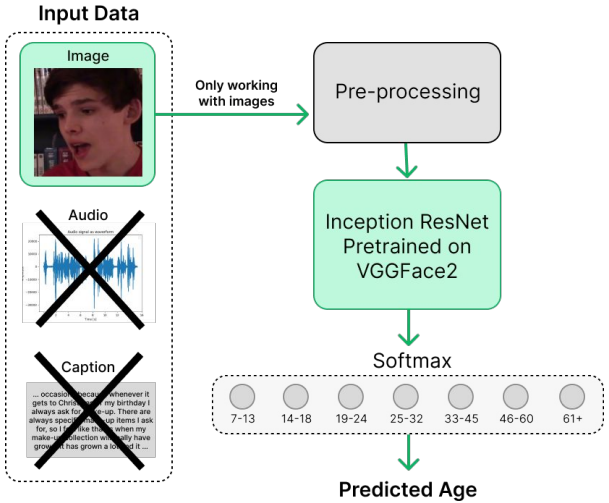
STRATEGY: **Late fusion.** We have trained each model separately and combined them afterwards.



Task g. Multimodal model.

Image Classification Model:

We have used the same model as in the task b, but this time with the new train implementation of the task c (data-augmentation + weighted loss). The model is an Inception ResNet model pre-trained on VGG-Face2. For the loss, weighted Cross-Entropy func is used to compute the logits, which are then processed through a softmax function to yield the final age predictions. The weights for each class are inversely proportional to the number of train images of each class.



Loss:	Weighted Cross Entropy
Optimizer:	AdamW
Learning Rate:	1e-6
Batch Size:	256

Image model Test Accuracy: **0.54**

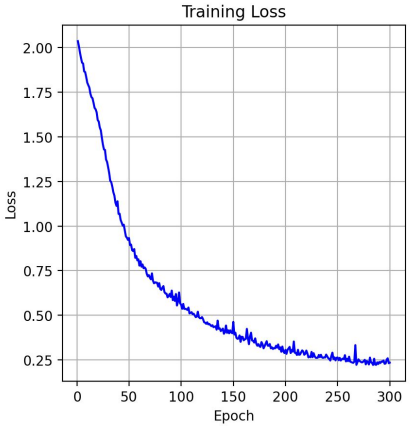
Task g. Multimodal model.

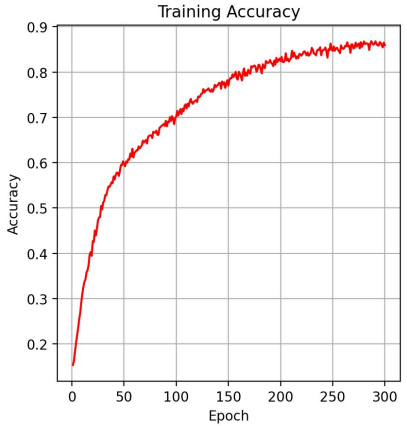
Image Classification Model:

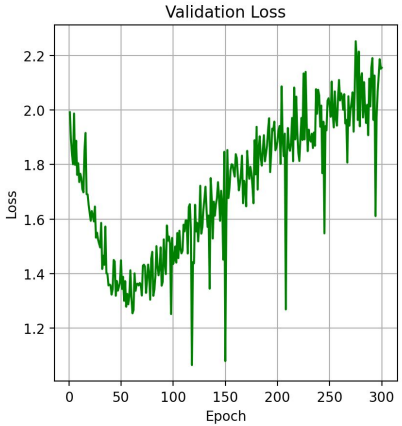
As we can see the in the validation accuracy plot, there is a plateau where the model generalizes well (epoch 50–100) but after that the model undergoes bad performance:

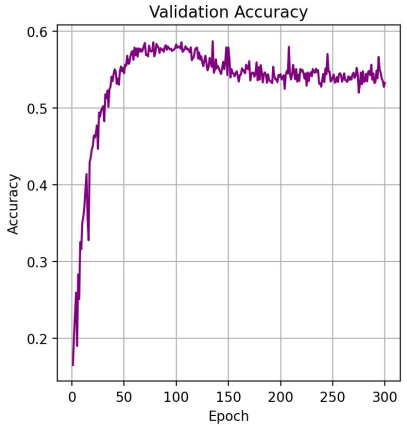
Loss:	Weighted Cross Entropy
Optimizer:	AdamW
Learning Rate:	1e-6
Batch Size:	256

Image model Test Accuracy: **0.54**







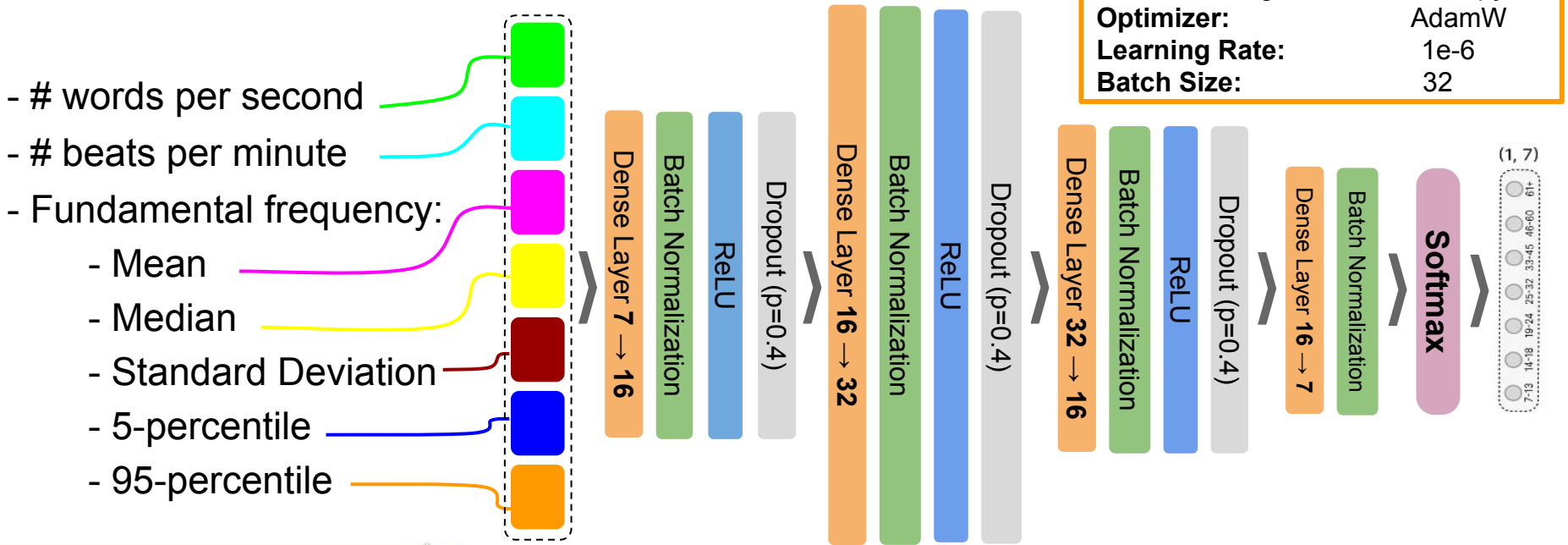


Task g. Multimodal model.

Audio Classification Model:

In this case we are not working with images, and neither with an embedding. We just have 7 values to work with for every audio, the ones extracted in task e). So, a quite simple linear model was created to classify the extracted features into age-ranges:

Loss: Weighted Cross Entropy
Optimizer: AdamW
Learning Rate: 1e-6
Batch Size: 32



Task g. Multimodal model. (task g) and e))

OBSERVED ISSUE: Audio feature extraction takes a extremely large time, **!! ~18s per audio !!**. The largest amount of the time is needed computing the ***fundamental frequency (f0)*** (~16s). Which makes the training and evaluating process too computationally expensive: **a !! single epoch needs ~31 hours !!** (using a *NVidia RTX 3090*).

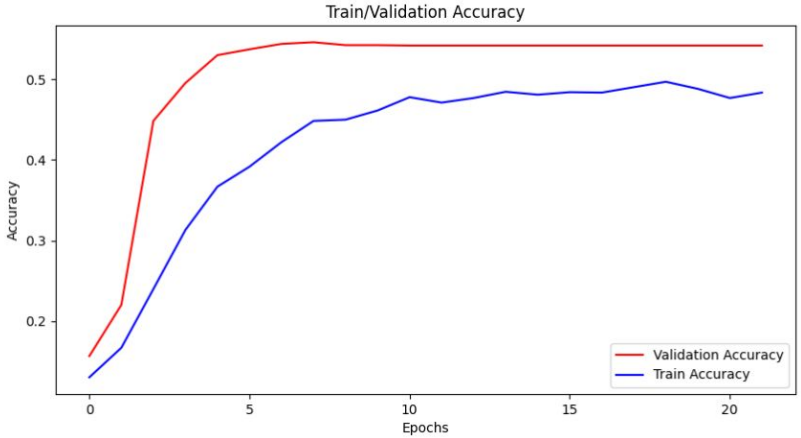
A rethinking of the strategy is necessary !

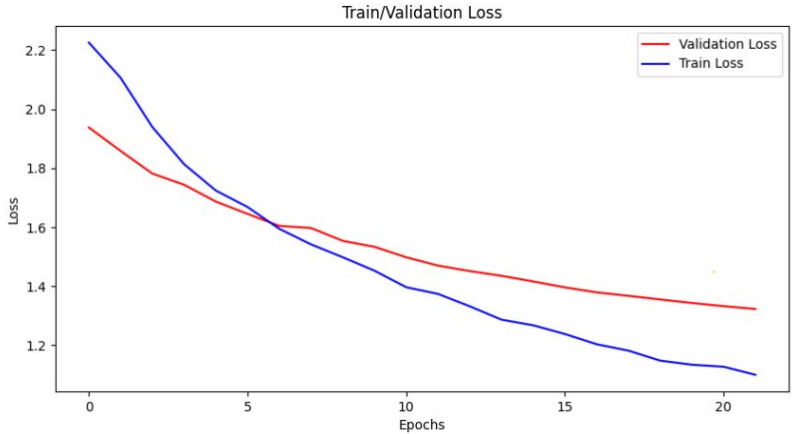
We can't get rid of the fundamental frequency, since we extract 5 of the 7 features from it, so the only way of going is to speed up the computation process:
The function ***librosa.pyin(data, fmin, fmax, frame_length=2048)*** is used to extract *f0,* and, by looking at the spectrograms (examples in the slide 25), we see that most of the fundamental frequencies are found below 500-700Hz. So, in order to shorten as much as possible the range of analysis, we have used ***fmax=450*** (instead of 2048, which was used at the beginning to retrieve as much information as possible).
On the other hand, we have also decided to augment the frame length, using ***frame_length=4096.***

TIME COMPARISON:
- **Before new strategy:**
~18s per audio,
~31 hours per epoch
- **After new strategy:**
~2.4s per audio,
~16 minutes per *(reduced)* epoch

This way, we have been able to reduce the **feature extraction of each audio from ~18s to ~2.4s**. Moreover, we have used **4 workers** to do more computations in parallel.
Finally, we have reduced the epoch lengths so we obtained faster accuracy and loss information, using **in each epoch just a random third part of the whole training set** (augmenting the # of train epochs and saving the checkpoint of the best epoch, obviously).

Task g. Multimodal model.





Audio model test accuracy:

49.37%

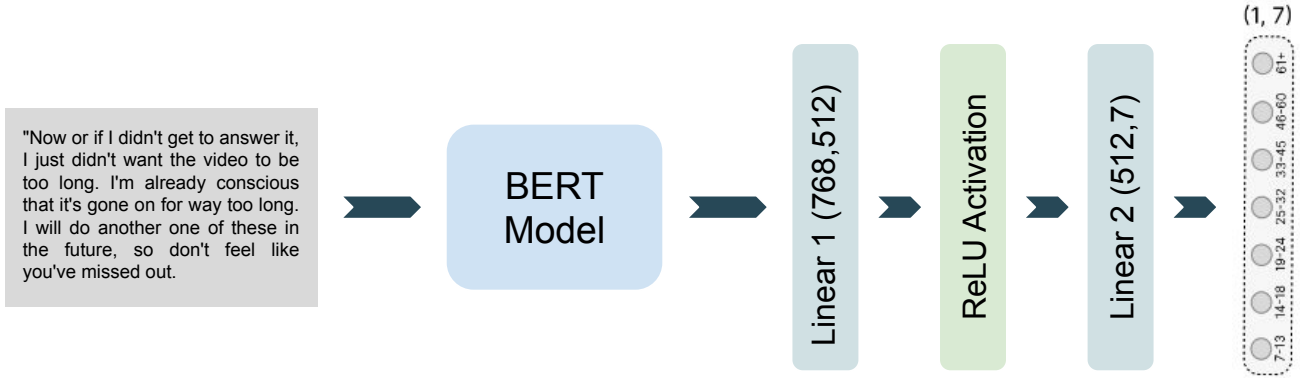
It can be seen how there is still some room for improvement for the audio model, even though it is not a lot, given that the validation accuracy is stuck.
But, since we did not have time, we have retrieved the checkpoint with the **lowest validation loss.**

The observed validation accuracy is higher than the train accuracy. This reflects the fact that **the model generalizes correctly**, and, from our perspective, with more time we could use the entire training set on each training epoch (instead of a random third part in each epoch), and so the model would be able to further improve its accuracy and loss (at least in the training set).

Task g. Multimodal model.

Text Classification Model:

As we have explained in task f, we have employed BERT in order to extract the deep text features of the user's transcriptions. Once we have the BERT's tokenizers we have created a very simply model composed by a 2 Linear layers and a ReLU activation function to obtain the label predictions:



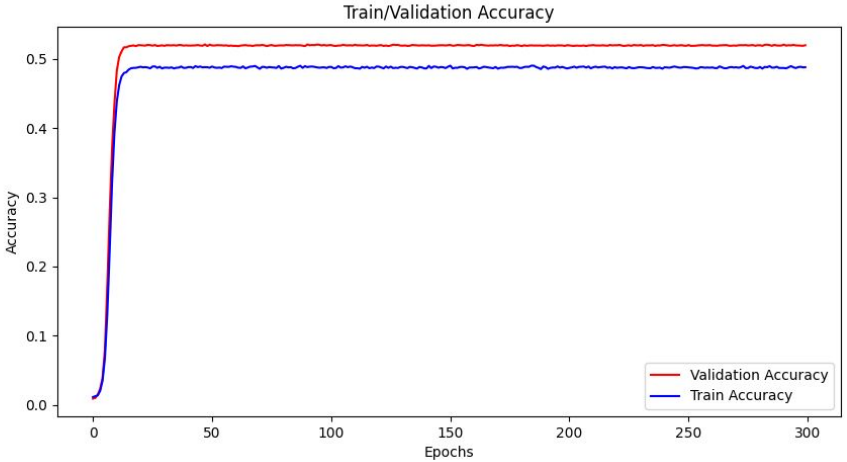
Loss:	Weighted Cross Entropy
Optimizer:	AdamW
Learning Rate:	1e-6
Batch Size:	256

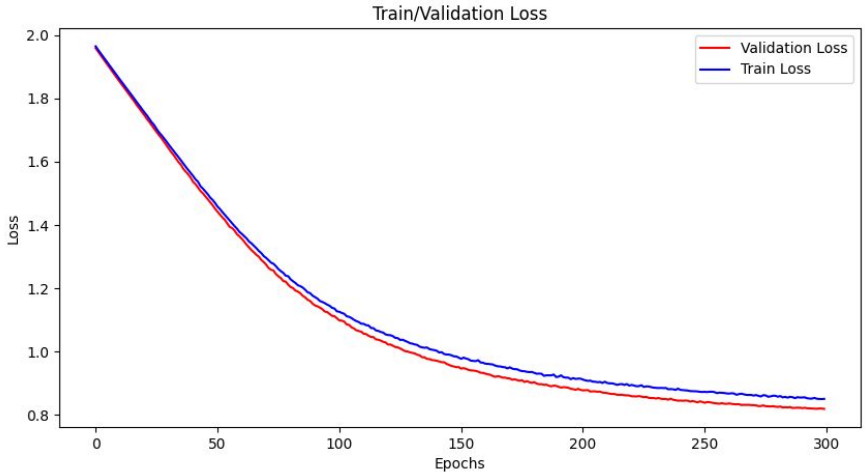
Text model Test Accuracy: **0.49**

Task g. Multimodal model.

Text Classification Model :

After training we can analyze a bit their accuracy and loss curves. The validation and training values are very similar so there is a limitation in the characteristics of the text data that prevents the model from learning to differentiate ages with only text. The maximum accuracy obtained is near 0.5, that is quite good if we take into a account that there are 7 different classes, and we are only using text features. Moreover, the accuracy stagnates around the epoch 15, but the loss continues to improve.

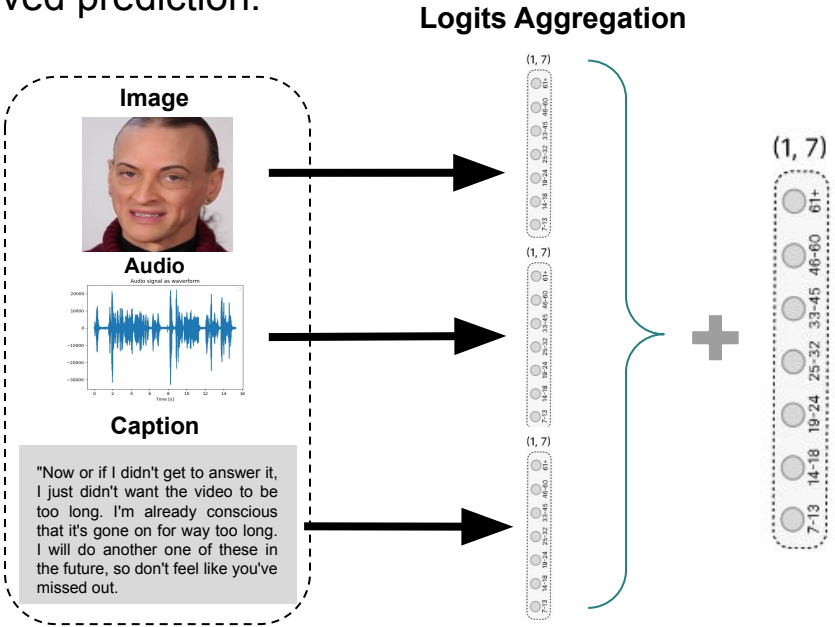




Task g. Multimodal model.

Combining Models:

Once the three models have been trained and saved their weights, we create a new inference model that **combines the three predictions of the same VideoName** to obtain a final and improved prediction.

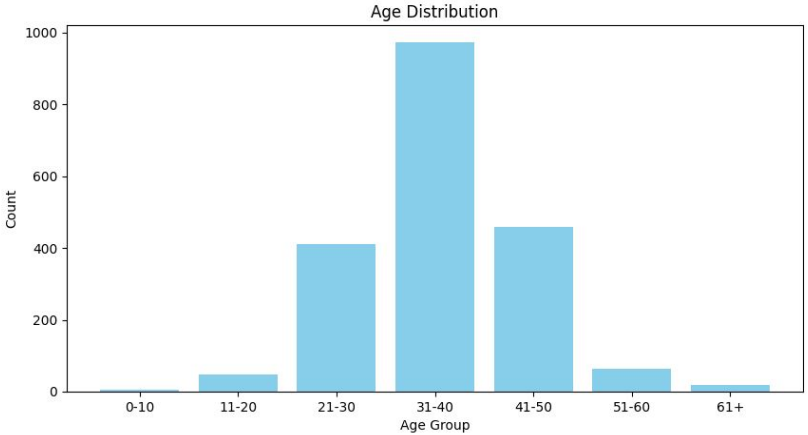


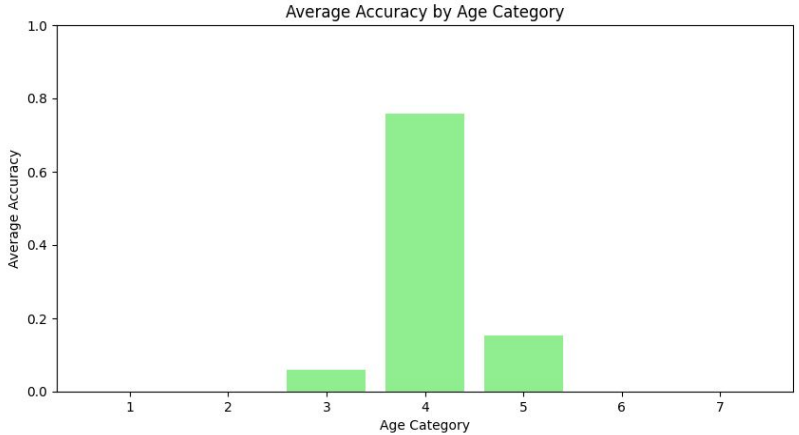
With the combination of predictions we have obtained an improved solution in terms of Test Accuracy. **We have outscored the results of task b** only using images, so we can confirm that **the use of different data types in a multimodal model is a way of improve our classification model.**

Image Test Accuracy: **0.535**
Audio Test Accuracy: **0.49**
Text Test Accuracy: **0.49**

COMBINED TESTING ACCURACY: **0.55**

Task g. Multimodal model.





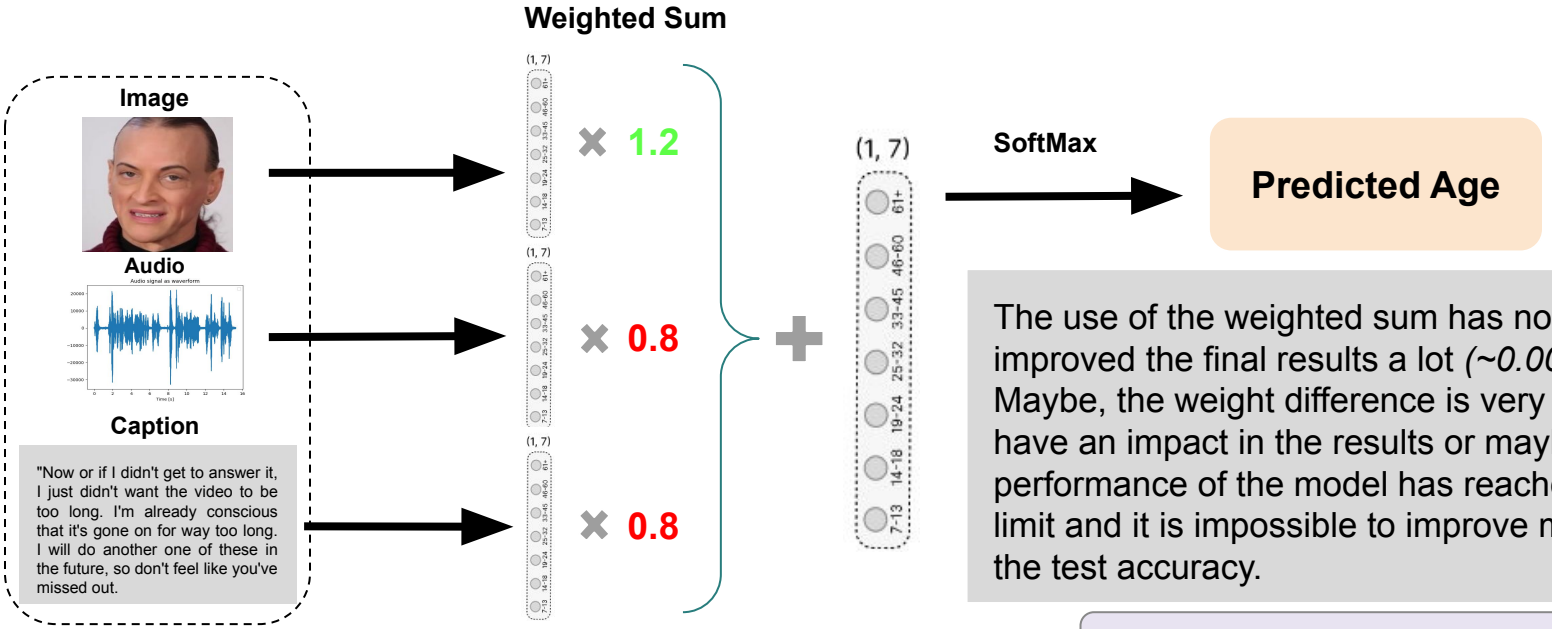
It can clearly be seen how **the data distribution imbalance plays an important role in the multimodal prediction accuracy per class**.
Even though we have tried to minimize the impact of the class imbalance, the multi-modal model is able to obtain an acceptable accuracy on just one class. Moreover, out of 7 there are just 3 classes which have a non-zero per-class accuracy.

This issue clearly points out the **importance of having a balanced data distribution**, otherwise, even when trying different strategies to solve this issue, the imbalance will have a high impact on the obtained results.

Task h. Ablation Study.

Task h. Ablation Study.

In order to know which type of data has more importance in the final result we can combine the different model solutions with a weighted sum, giving different weights to each data type result. In the previous task we have seen how the image model outscored the other models, so we propose to give a higher weight to the image results than to the others:

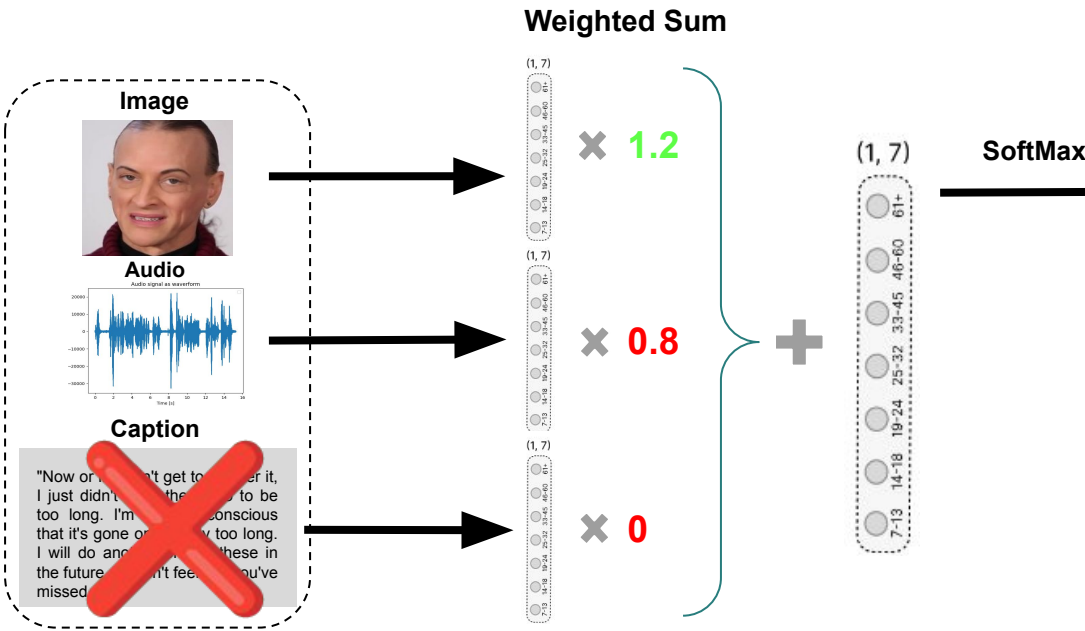


The use of the weighted sum has not improved the final results a lot *(~0.001)*. Maybe, the weight difference is very low to have an impact in the results or maybe the performance of the model has reached its limit and it is impossible to improve more the test accuracy.

TESTING ACCURACY: **0.55**

Task h. Ablation Study.

We think that the use of text in an age classification problem is a very hard task and it is very difficult to know the age of a person only using the words that he/she uses in a 15 seg video. So we are going to run the combined inference model, but this time **only using image and audio**, to know if the text has importance in this task or only is introducing noise.

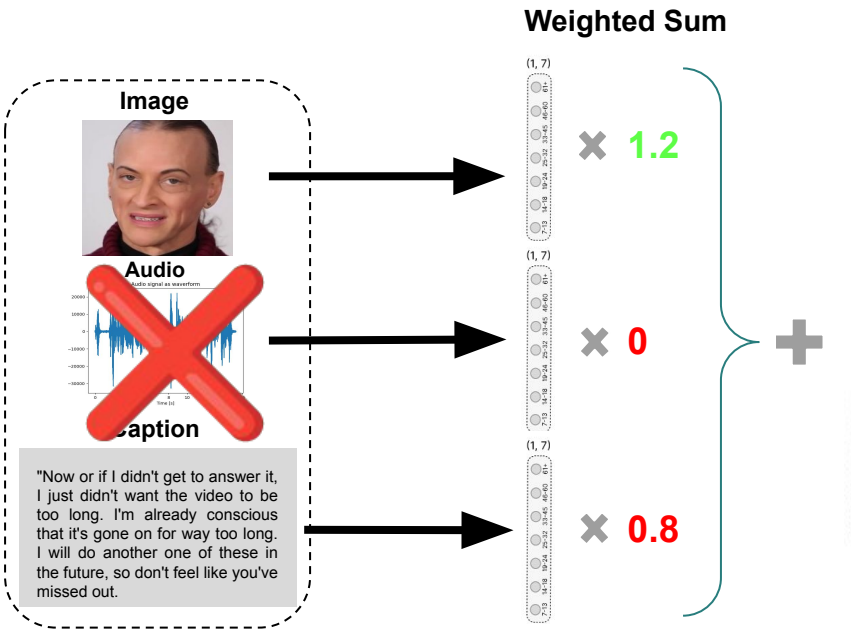


After proving this new method the results are a bit lower than the previous one using the text. So, the text does have a positive influence on the results and it is important to include it in the multimodal model implementation.

TESTING ACCURACY: **0.54**

Task h. Ablation Study.

After proving only the combination of image + audio and obtaining lower results, we have to try the multimodal model but **without audio** (image + text). Using this combination we can check the importance or not of the audio in this classification task.

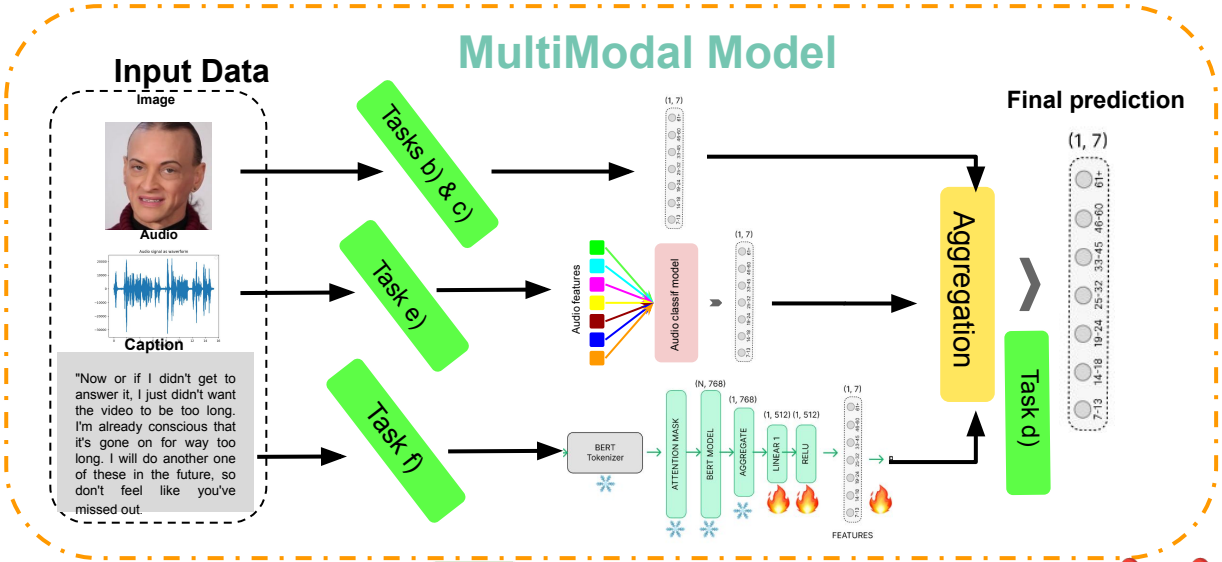


Again, the combination of only two data types gives us lower results than the full combination. In this case the results are a bit lower than only image + audio, so the importance of the audio data is higher than the text, confirming what we expected.

TESTING ACCURACY: **0.54**

SUMMARY SLIDE

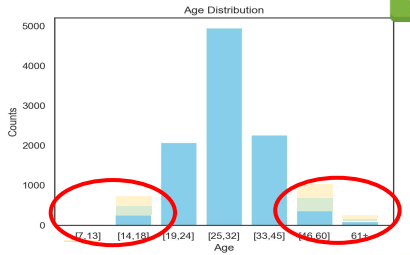
SUMMARY SLIDE G7



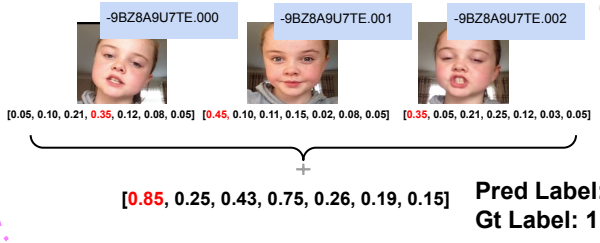
Individual model results:

- **Image model accuracy:**
  Train: 85.96% Val: 58.71% Test: 53.50%
- **Audio model accuracy:**
  Train: 51.29% Val: 54.18% Test: 49.37%
- **Text model accuracy:**
  Train: 48.98% Val: 52.0% Test: 49.37%

Train Improved:



Test Improved:



Test Accuracy Task b: 53.50%
Aggregating Logits: 54.89%
Weighted sum: 54.90%
Image + Audio: 54.42%
Image + Test: 54.12%
High impact of data imbalance !!