# Guess The Age Contest: Gruppo 12

Ferdinando Sica, Vito Turi, Mattia Marseglia, and Camilla Spingola

{f.sica24, v.turi3, m.marseglia1,
c.spingola}@studenti.unisa.it

## 1 Introduction

Estimating the age from faces is a very challenging task as there are numerous aspects that can influence this choice such as ethnicity or gender. Moreover, the faces of people of the same age may have a very different appearance due to the characteristics of the person represented because of the uncontrolled ageing process or due to image's characteristics such as pose, lighting or quality. Likewise automatic age estimation from face images has numerous practical applications such as demographic statistic collection, customer profiling, assistance of biometrics, etc.

In the competition it has been made available the MIVIA Age Dataset, a very large Dataset composed of 575.073 images of more than 9.000 identities of different ages, very heterogeneous in terms of face size, illumination conditions, facial pose, gender and ethnicity. It is important to underline that the dataset distribution is not homogeneous for the different ages and that there are very few samples for the external ages, reproducing a tailed distribution.

The model we propose to take part in the contest has been designed being aware of the complexity of the task to be carried out and of the problems that would arise from the distribution of the dataset. Another central point of the design has been to obtain a performing product that did not require excessive resources and above all that guaranteed the explainability property of the architecture.

For all these reasons we design an architecture made of different weak learners, each one with a specific task and consequently with a specific architecture and trained in different times, using as features extractor the InceptionResnetV1 starting from already trained weights and making fine-tuning on it. The ensemble of the weak learners is all about to obtain a stronger learner with more varied functionality.

## 2 Description of the method

Age estimation from faces is not strictly a classification problem since it foresees the possibility of having continuous output even if in a defined range (for example, the age ranges between 1 and 100 years can be considered valid). Obviously, it is possible to deal with this problem as if it is a pure classification, considering a number of classes equal to the different allowed age values. In the same way hybrid methods or methods built ad hoc for this problem could be applied, such as for example the Reduction Framework.
In choosing how to deal with this problem, all these possibilities were taken into account to, then, make a selection with respect to the available dataset, the specifications provided to us on the maximum size of an ensemble and the available computational and temporal resources.
Specifically, the dataset distribution influenced the choice of the architecture by discouraging a pure classification approach with a number of classes equal to the different ages. In fact, the dataset is really unbalanced in the number of samples for the different ages and so, creating a pure and simple classifier would have been too difficult, even predicting data augmentation approaches.
The specifications about the maximum size of the ensemble equal to three discouraged methods like Reduction Framework because it requires at least k-1 binary classifiers where k is the number of different age groups.
Starting from these considerations we choose to face this task with a regression approach. Of course, regression problems are generally more complex to handle than classification problems due to the continuous value of the outputs. Therefore, we made use of an available fairly large dataset and the possibility of using an ensemble to smooth out the pitfalls of a regression.

### 2.1 General architecture

The general architecture of the proposed model is a hybrid architecture.
The architecture is composed of an initial backbone, two classifiers and a regressor.
The backbone is the network InceptionResnetV1, pretrained on the VGGFace2 dataset, from which we clone the layers, starting from the "mixed_6a" block to the end, in order to obtain two heads. One of these heads is tied with two classifiers, while the other is tied with a regressor.
The head of the backbone was split between the regressor and the classifiers so that each one of the single experts can fit the weight of this part according to the specific task it is used for.
Two classifiers share the weights of the part of backbone from block "mixed_6a" to block "mixed_7a" (*Final Backbone Part for both* Classifiers in the image below).
The task of the two classifiers is, starting from the features extracted by the block "mixed_7a", extracts more adequate features to classification task, using both remaining backbone layers (*Final Backbone Part for 1st or 2nd Boosted Classifier* in the image below) joined with new added layers (*1st or 2nd Boosted Classifier Part* in the image below) and then predict the age of the person solving the problem as it is a classification one. By a specific combination rule, which involves the elaboration of classifier's outputs, the outputs of the two different classifiers are concatenated together into a final output vector.
The head tied with the regressor has the task of, starting from the features extracted by the common part of the backbone, extracting more adequate features to regression task. Then regression's features are concatenated to classifiers' final output. This concatenation is passed as input to some added final layers (*Regressor Part* in the image below) of the regressor which, starting from the extracted features and the information content of the classifiers' output, must predict the real age.

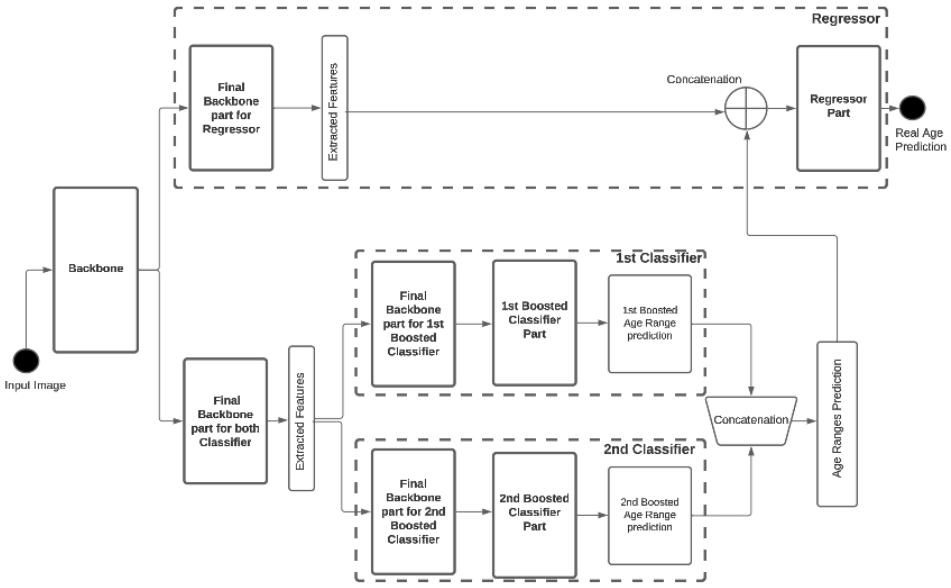


*Fig 1. General Architecture Scheme.*

**Backbone**

To extract the features from the images we use an already trained network as backbone, in particular the InceptionResnetV1 trained on the dataset VGGFace2. VGGFace2 is a dataset made of around 3.31 million images dedicated to people's identity recognition. It has been used to train different networks for face-related tasks, because it contains images of different poses and age diversity. So hopefully pretraining on this dataset makes the network capable of having a good ability to extract the most disparate features starting from a face's image. All these reasons made the pretraining on this dataset a suitable choice for our problem, also because clearly, we cannot use a network already trained for our specific task of age estimation. For our time and resource limits we could not train from scratch a network on the face recognition task using the VGGFace2 dataset, so we chose among the pre-trained models available for PyTorch Framework. Among all of these, we choose the model InceptionResnetV1, whose pretraining is available on a GitHub repository also because it is not a very deep network (it contains 27.910.327 total trainable parameters), and so give use guarantees of satisfying our computational and temporal constraints during the training phase, even with layers added by us and with the proposed architecture. Regarding the prediction phase, it also gave us guarantees of being able to carry out the prediction in a limited time. On this aspect, although not explicitly requested, we felt it was our duty to pay attention, foreseeing a possible use of the implemented system in a real context.

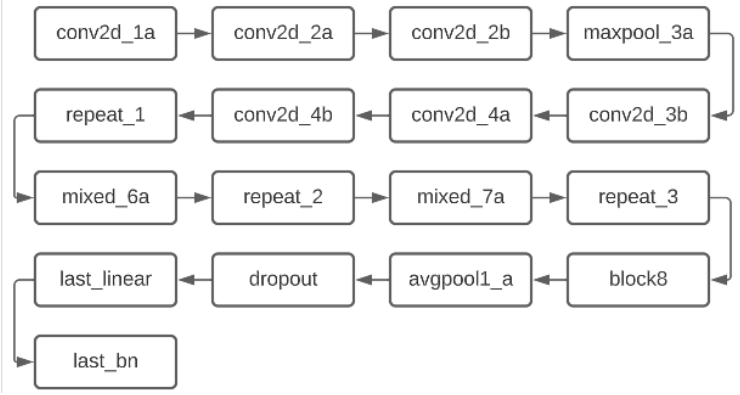


*Fig 2. InceptionResnetV1 architecture.*

**Single experts**

The single experts used are two classifiers and a regressor, each one with a specific objective. Since these are different experts, clearly, each one has its own architecture and for each one a particular training procedure has been prepared with a specific loss in order to maximize their learning.

***1st Classifier and 2nd Classifier.*** The two classifiers represent the middle of the model. Their task is to predict the age label of a person. Both classifiers were designed for a classification with 81 classes, corresponding to the possible ages allowed by our system. The use of these two classifiers serves as a support to the regression task, making it less complex, so basically the idea is to further reduce the interval in which to do regression for each sample. In fact, both classifiers have been trained considering the output as the argmax on the Softmax's resultant vector, but this output was processed, before giving it to regressor, obtaining a range of ages containing the predicted one. The two classifiers have the same architecture as it is possible to see below, with some blocks corresponding to the least part of the backbone of which some layers shared the weights, and some others have personal ones and finally there are some layers added from scratch. The layers added from scratch include different configurations of Linear, ReLu, Dropout and Batch Normalization layers.

ReLu is used as an activation function to avoid the problem of vanishing gradient. Batch Normalization layers are used to normalize the input of the layer and to maintain the mean output close to 0 and the output standard deviation close to 1. Dropout layers are used to avoid overfitting, but the percentage has been set to low values to not make the convergence time excessive.

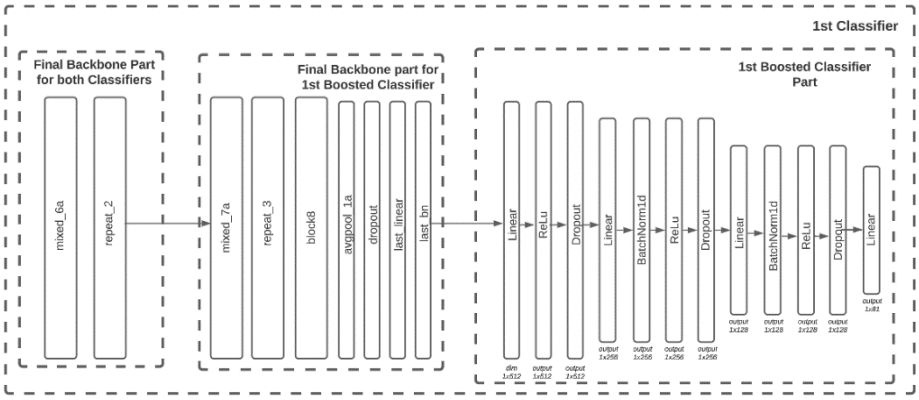


*Fig 3. 1st Classifier Architecture.*

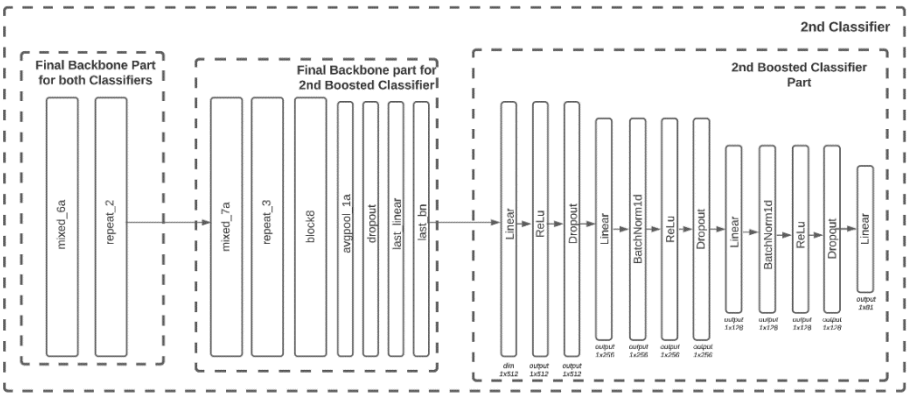


*Fig 4. 2nd Classifier Architecture.*

What is different between the two classifiers is the applied training procedure which justifies the choice of having two different classifiers even if with the same architecture. In particular we have decided to apply a boosting procedure to training the two classifiers. The First Classifier learns on the entire training set, considering all

the samples with the same weights equal to one. The Second Classifier learns on the entire training set, so no bagging procedure has been applied, but unlike the first classifier, it considers with higher weights the samples misclassified by the first one and with lower weights the samples correctly classified by the first one.
The use of the two classifiers stems from the idea, as will be better explained below, of improving performance, making the two learners as much complementary as possible to each other and proposing a first palliative to the imbalance of the dataset, together with different cost for misclassifications on less numerous classes.
Obviously, age estimation is an anomaly classification problem, because misclassifications do not have the same weights, since, according to the classes among which the mistake is made, the error is about a greater or a smaller number of years, so also this aspect have been taken into account during the training procedure as better explained in training procedure section.
Among the possible classifier's architecture, surely one could have been the classification with a number of classes lower than 81, and so for example grouping the ages in ranges of ten of fifteen years. As will be reported below, this approach has been tested, but it involves the fact that prediction errors are made on adjacent classes most of times misclassifying for only one years on "edge samples", that are very similar to each other, but it cannot be taken into account during learning procedure, and so this type of approach makes the classification task extremely more difficult.

***Regressor.*** This system is necessary to predict the real age. As it is possible to see in the image below this regressor is composed of two parts, the first one that is the final part of the backbone trained for regression task and the final part "Regressor part" that are some layers added from scratch. This last part takes in input the concatenation of the feature vector extracted from the final part of the backbone and age ranges prediction coming from classifiers' outputs combination. This structure has some layers made from scratch and contains three different types of layers: Linear, Average Pooling and Dropout. Average Pooling layer are used to reduce the spatial size of the data and ensure a certain degree of translation invariance in fact in this way a small change in the position of a detected feature should not affect the final result of the network. Dropout layers, that are equivalent to temporarily "shutting down" the neurons with a certain probability, are used to reduce the variance error in order to reduce the overfitting. So, the use of the dropout layer has been equivalented to training many small networks and combining their output by averaging test/usage time. Obviously to predict the age the last layer is a Linear that outputs a single value. So autonomously the regressor extracts from the input image the features necessary to carry out the face analysis, and then combines them with the information content obtained from the classifiers. This informative content should allow him to narrow the age range within which to predict the age. So, it is necessary that the "Regressor part" represented in the architecture below is able to interpret the features coming from the images and also the information coming from the classifiers.

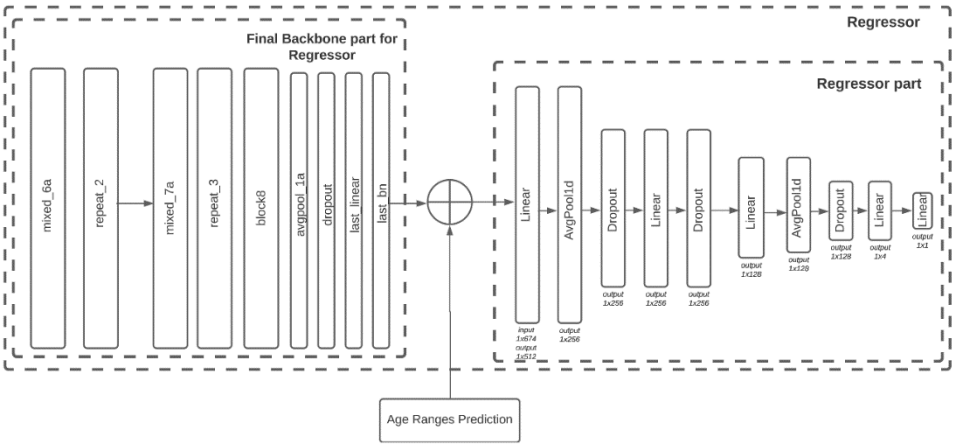


*Fig 5. Regressor Architecture.*

**Multi-expert system**

As previously mentioned, the proposed architecture for the contest is the result of a hybrid approach, because it combines regression and classification. The presence of the two classifiers has two main objectives: the first is to make regression task simpler reducing the range within which to predict the continuous age value; the second is to propose a mitigation to the dataset imbalance. About the second purpose, as described above a boosting technique has been applied in order to improve the general performances and also to ensure that the more difficult samples on which the first classifier fails, which reasonably are those of the least numerous classes, are considered with a higher weight from the second classifier. Among the different methods that can be used to explore the problem of class-imbalance we choose this approach, that seems to be more stable respect to bagging techniques that include undersampling or oversampling the dataset randomly, avoiding the removal of samples that are very representative of the class or the addition of synthetic samples that are not truly representative of the class. About the first purpose, and so the simplification of the regression task, the output of the two classifiers is processed and also given as input to the regressor. In particular, starting from the output of each classifier, through SoftMax, are obtained the probabilities that a sample belongs to each class, then is computed the argmax on these probabilities and this corresponds to classifier's output. Then the age range predicted by each classifier is the range of four years before and after the age corresponding to the argmax above mentioned. Then for each classifier is created a vector of 81 values, all zeroes except for the nine (4+4+1) indices corresponding to the obtained range. The vectors of each classifier are concatenated, first that of the First Classifier and then that of the Second Classifier, obtaining a total vector of 162 values. We decided to assign to the output of each classifier a vector of 81 cells, and not a smaller number, in order to manage all the possible age range values. Furthermore we decide to not combine the age group vectors of the classifiers with a particular combination rule, but simply concatenate them in order to leave the management of this information content entirely to the "Regressor part", so that on the basis of experience and of the extracted features, it could understand whether to believe more in the age range contained in the first half or in the second half of the vector obtained by classifiers. The choice of use a range of four years before and after the predicted age was dictated by an analysis (reported below) of the performances of the combination of the two classifiers in different ranges, in

fact four allow to achieve satisfying performances for our purposes even without including an excessive wide range, maximizing their complementarity. Finally, the regressor must predict the real age of the samples. Obviously, from the point of view of an ensemble, the regressor must benefit from the job done by the classifiers that precede it, but also perform its own inference on the sample to predict the precise age. This motivation justifies the concatenation of the feature vector and the prediction of classifiers to compose the regressor's input.

## 2.2 Training procedure

**Dataset**

The age estimator was trained using the MIVIA Age Dataset, which was provided for the contest. The dataset consists of over 9000 identities with a total of 575,073 images taken at different ages.

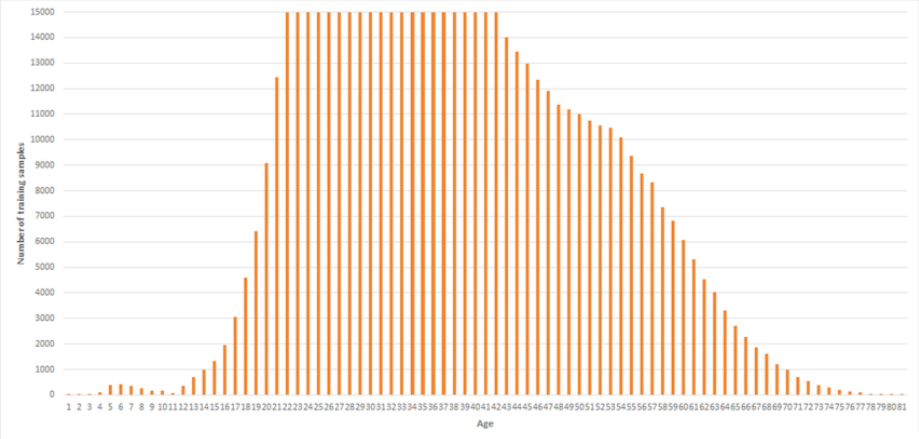


*Fig 6. Dataset distribution for each age.*

These images were extracted from the VGGFace2 dataset and annotated using the "knowledge distillation" technique, resulting in a heterogeneous dataset with a wide range of face sizes, lighting conditions, poses, genders, and ethnicities. The images provided in the dataset are already cropped to show just the face, with different sizes.

As shown in figure above, the distribution of ages in the dataset is heavily imbalanced, with some ages (at the tails of the distribution) having only 2-10 images per age. To further divide the dataset into a training set and validation set, we used a random splitting technique, where we randomly assigned 74% of the images to the training set and the remaining 26% to the validation set. The split was done in a way to respect the distribution of the entire dataset.

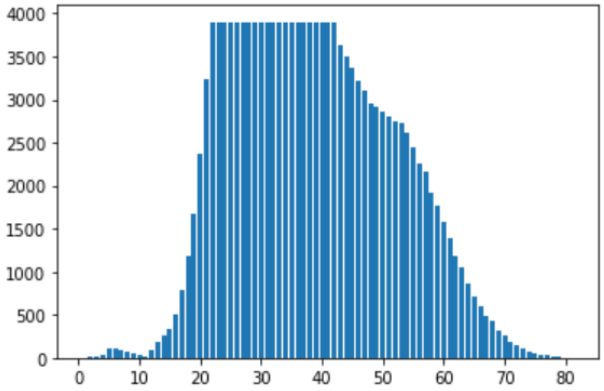


*Fig 7. Validation set distribution for each age.*

This ensures that the training and validation sets are representative of the overall dataset and gives us a sense of how well the model is able to generalize to new, unseen data. Dividing the dataset into a training set and a validation set is a common practice in machine learning, and it is especially important when we have a conspicuous dataset. By using a validation set, we can get a more accurate assessment of the model's performance and make sure that it is not overfitting to the training data. Splitting the dataset in this way can also help us to tune the hyperparameters of the model more effectively, as we can use the validation set to evaluate the impact of different hyperparameter choices. Overall, using a training set and a validation set can help us to build more robust and generalizable models.

To evaluate the performance of the model on the validation set, we used the AAR (Age Accuracy and Regularity) metric as described in the contest rules.

$$AAR = \max(0; 5 - mMAE) + \max(0; 5 - \sigma)$$

We have been inspired by this metric to assess the effectiveness of the model and eventually compare it with others. A perfect method will achieve AAR = 10. This metric is composed by these two parameters:

$$\sigma = \sqrt{\frac{\sum_{j=1}^{8}(MAE^j - MAE)^2}{8}}$$

$$mMAE = \frac{\sum_{j=1}^{8} MAE^j}{8}$$

**Face pre-processing**

For the pre-processing phase we decided to use a face alignment technique. In particular, face alignment refers to the process of identifying and locating key points on a face, such as the corners of the mouth, the tip of the nose, and the center of the eye sockets. These points are then used to align the face in a standardized way, so that it is oriented in the same position relative to the image, and then those points are always in the same locations, regardless of the pose or expression of the person in the image.One common technique for performing face alignment is to use a combination of feature detection and geometric transformations. Feature detection algorithms are used to identify the key points on the face, and then a geometric transformation (such as a similarity transformation or an affine transformation) is used to align the face based on these points. Here is an example of an image before and after face alignment:

*Fig 8. Image before alignment.*

*Fig 9. Image after alignment.*

As you can see, the second image has been aligned so that the face is centered and oriented in a consistent way, which can be helpful for many downstream tasks such as face recognition and facial expression analysis. Before the process of face alignment, the images were resized to 160x160 and, after the process, converted to Tensors. We used this size for the images because the backbone of our system was trained on images of this dimension, so we decided to maintain the same size in order to obtain the same high performance it had on VGGFace2. The Tensors were then normalized by subtracting the mean and dividing by the standard deviation in order to reproduce the original training procedure. This normalization helps to improve the performance of many machine learning models by scaling the input features to a similar range and reducing the influence of any outliers.

**Data augmentation**

Data augmentation is a common technique used to increase the size and diversity of a dataset in order to improve the performance of machine learning models. There are many different ways to augment data, and the specific policies that are used can depend on the characteristics of the dataset and the goals of the model.

In the case mentioned, the data augmentation policies used were horizontal flipping, random rotation, and brightness variation. These policies were chosen because they have the potential to introduce diversity into the dataset without significantly altering the original images. For example, horizontal flipping can introduce symmetry into the dataset, while random rotation can help the model to learn to recognize faces at different angles, this should manage an eventually mistake in the alignment procedure. Brightness variation can also help the model to learn to recognize faces under different lighting conditions. Furthermore, light conditions can definitely influence the physical appearance of a person and affect the visibility of certain facial features. This can make it more challenging for a model to accurately predict the age of an individual based on their appearance.

By applying these augmentation policies to the original images on-the-fly during the training, the dataset can be augmented to include more diverse and representative samples. This can help the model to learn more robust and generalizable features, which may improve its performance on new, unseen data.

**Training from scratch or fine tuning**

In our case, we decided to use a pre-trained model called InceptionResnetV1, pretrained on VGGFace2. We then fine-tuned the model by modifying part of the layers of the network to better suit our task. This is a common approach when using transfer learning or fine tuning, as the last layers of the network are typically the most task-specific layers and that's right that they are modified to better fit the needs of the new task. The same procedure was also applied for the classification tasks. More specifically, the backbone is blocked from the first layer to the block "mixed_6a"; so, we trained the backbone from this onwards to fit those layers to our tasks.

**Training of the single experts**

In the contest the score was based on a specific performance metric called the AAR (Age Accuracy and Regularity). When the model is perfect in its prediction, the AAR value reaches a value of 10. However, given the specific structure of the AAR calculation, we found that minimizing mMAE and sigma is equivalent to maximizing the AAR value.

The mMAE component takes into account the regularity of the error, while the sigma component takes into account the regularity of the error across different age groups. By minimizing these two components, we aimed to improve the accuracy and regularity of the model's age predictions. So, for the regression task, we calculate the loss as mMAE + sigma. For the classification part of the network, we used the Pytorch Cross Entropy loss function with custom multiply factors. The purpose of the custom factors was to make the classification error personalized for the specific

problem. For example, using custom weights, we can help the model to focus more on certain classes and improve the overall performance on the age estimation task. Specifically, the losses implementations for classifiers and regressor are discussed in chapter 3. By using these two loss functions, we were able to train the different parts of our network to better fit to our specific task and improve the accuracy and regularity of the model's age predictions. The classifiers loss function will take into account the error weight of the first classifier to perform the boosting procedure with the second one.

An early stopping and checkpoint mechanism is a common technique used to improve the training of machine learning models. The early stopping mechanism allows the training to be interrupted when the performance of the model on a validation set stops improving, in order to prevent overfitting. The checkpoint mechanism saves the weights of the model at each improvement of the performance, so that the model can be restored to a previous state if the performance starts to degrade, we can also restore the model to a state where it had previously performed well.
For the classifiers and regressor, the early stopping was based on the loss value of the epoch. The same metrics were also used for saving the weights of the model.

We chose to use SGD as the network optimizer because it is a simple and widely-used algorithm that is well-suited for many machine learning tasks. The learning rate of 1e-3 and the momentum of 0.9 were chosen based on empirical results and previous experience and were found to work well for our specific task. SGD is a popular optimization algorithm that iteratively adjusts the weights of the network based on the gradient of the loss function with respect to the weights. The learning rate determines the size of the adjustment made to the weights at each iteration, while the momentum helps to smooth the updates and improve convergence.

We did different steps of training for the classifiers and the regressor. First of all, a first large training was made, in which we train the backbone from the block “mixed_6a” onwards joint with the layers added from scratch for the classifiers; so the first part of the backbone, as we said before, will remain blocked. After this first step of training, we loaded the obtained weights and used it to train the two boosting classifiers from the already trained one, each of them updated its own weights from the block “mixed_7a” onwards. In this way, with the first step of training we let the backbone to better fit to a new task, different from the one on which it was pre-trained (face recognition); with the second step, we fixed the weights of the two blocks “mixed_6a” and “repeat_2” and train 8 million of parameters less than the first step, where the parameters were 22 million. Obviously, this last part, without the two mentioned blocks, constitutes the head tied with the classifiers.

After the training of the classifiers, we trained the regressor structure. A first large training was made, in which we train the backbone from the block “mixed_6a” onwards joint with the layers added from scratch for the regressor (for a total of 22 million of parameters). Obviously, the part of the backbone trained with the regressor constitutes the head tied with the regressor layers and have different weights from the classifiers one. In a second training we load the pre-trained regressor model, we fixed the weights of the two blocks “mixed_6a” and “repeat_2” and trained from the block “mixed_7a” onwards.

The number of epochs for the training of single experts is 30; so, we trained totally 60 epochs for each learner. Obviously, we used early stopping so the number of training epochs has been chosen because we saw that increasing that number, the performance did not change, so a sort of stability is reached with these epochs.

**Training of the multi-expert**

The two classifiers have been trained with a boosting procedure through AdaBoost implementation. The AdaBoost procedure is a popular machine learning algorithm that improves the performance of other algorithms by constructing a strong classifier from multiple weak classifiers.

In AdaBoost, weak classifiers are iteratively fit to the training data by re-weighting the data to give more importance to samples that were misclassified in the previous iteration. After each iteration, the weights of the samples are updated to give more weight to samples that are still misclassified.

The AdaBoost algorithm calculates the weighted error rate of the current classifier at each iteration and updates the sample weights accordingly. The final prediction of the AdaBoost classifier should be obtained by combining the predictions of all the weak classifiers using some combination rule but we choose to use the regressor to perform it.

The training of the final multi-expert system involves fixing the entire architecture of classifiers and the *final backbone part for regressor (in the Fig.1)*, while leaving the "regressor part" (in the Fig.1) free to adjust the weights. It is important to notice that for the “regressor part” we load the already trained weights, whose training procedure has been described above, only for the first layer of this section it was necessary to start with random weights. This because, the first layer of the “regressor part” that is going to be used will be slightly modified from the one used in the early training phase. The modification involves expanding the input layer of the "regressor part" by expanding the input size of 81 * 2 values to handle the concatenation with the output of the two classifiers. The added values will allow the regressor to process and incorporate the predictions made by both classifiers. The regressor receives the features extracted from the network and the predictions from the two classifiers rearranged as a tensor that contains the age ranges, as said before. The regressor acts as a combinator that maps the outputs of the base classifiers, together with the extracted features from image to the final output. This approach allows for a more sophisticated combination of the predictions of the individual classifiers, compared to simple averaging or voting, as the regressor can learn to give more weight to classifiers that are more accurate and less weight to classifiers that are less accurate, also depending on the evaluation on the available image features. This can lead to improved accuracy compared to simple averaging or voting, especially when the individual classifiers have different strengths and weaknesses.

# 3 Loss function design

As described above, because of the fact that the learners which are part of our architecture are different in tasks and structure, two different loss function have been designed, one for the classifiers and another for the regressor, in such a way as to make the most of their potential and maximize their learning.

First of all, obviously, the choice of the loss functions to be used has been influenced by the goal to be achieved and by different reflections on the dataset.

The dataset provided to us, in fact, is very promising in quantity and in quality in relation to the reality of interest, but it is largely unbalanced. It is dangerous if the goal is, as in our case, to obtain homogeneous performances at different ages. In fact, the consequence that this imbalance could bring was to obtain better performances only for the classes with more samples.

In relation to this aspect, it is important to note that the imbalance problem will be hidden by loss as MAE or MSE for their intrinsic predisposition to evaluate the average error made. So, about loss, our goal was to cross the imbalance of the dataset as much as possible, achieving little variability in the age prediction errors between different age groups. However, the actual prediction error should not be overlooked, otherwise a loss that only takes into account the error variability could be zero even when the errors are similar between different classes but very large in value. So, because of all these considerations, the designed losses take into account for the minimization of both of these factors (variability error and prediction error), together with the fact that there is an ordinal relation between different ages (especially for the classifier).

## 3.1 Classifiers loss function

The two classifiers have the same loss that starts from PyTorch Categorical Cross Entropy, but then take into account many aspects of customization of the problem.

First of all, the imbalance of the dataset is taken into account, associating different weights to different classes inversely related to the number of samples. During this phase an important aspect has been considered. In particular, in age estimation, even when faced as a classification problem, there is a strict relation between samples belonging to adjacent classes. Therefore, the error that could be made on each class in relation to the prior distribution must take into account also the number of samples of the adjacent classes, which in any case represents a helping factor. So, because of this reason, to compute the weight, we have made a sort of Label Distribution Smoothing technique (LDS). Specifically, what has been done was to obtain the prior distribution of the training set, then perform a convolution with a symmetric kernel (a gaussian one) in order to obtain a smoothed distribution that most likely represented the distribution of the errors on the various ages starting from the prior distribution of the data.

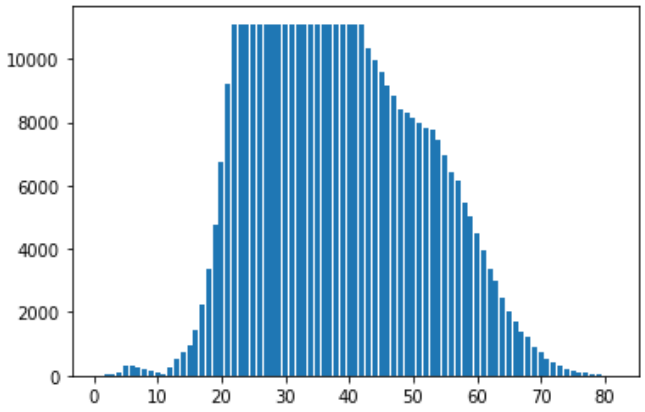


*Fig 10. Prior Distribution of the Training set.*

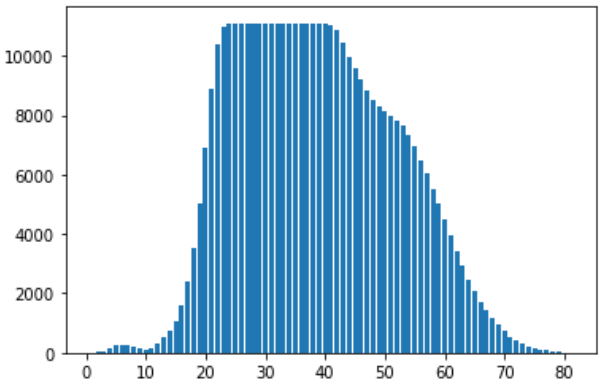


*Fig 11. Error distribution based on smoothed prior distribution.*

As it is possible to see in the graphs above, the error distribution is smoother than the prior one, creating a less accentuated disparity between the classes.
After this, we have done a sort of flattening of the weights to mitigate their imbalance. The need to flatten this disparity comes from the fact that it is not the only palliative used to mitigate class imbalance. This flattening technique takes into account the mean of the weights between the different classes and calculates the distance of each weight of every class from it. This distance is used to flatten the weights, approaching them to the mean; we approach the weights by 40% of their distance from the mean. After this, the weights are normalized for the minimum of them, obtaining a new distribution of them that preserves the initial ordering between classes, but with a more flattened ratio. The smoothed and flattened weights are finally ready to be used for the Categorical Cross Entropy loss. The value returned from the Categorical Cross Entropy is then multiplied for the weights of that specific sample. This aspect is the difference from the first and the second classifier, in fact the first classifier weights all the sample with a value equal to one, the second classifier, instead, take into account weights updated after the boosting procedure. The purpose of the boosting procedure applied in fact is to address the issue of class imbalance. It works by assigning higher weights to the samples that were misclassified by the first classifier, which is expected to be more complex. In this way, the second classifier is able to focus more on these samples and correct the errors made by the first classifier, thus improving the overall performance of the model. So, the loss will be equal to the value of the Cross Entropy Loss, calculated with suitable weights for each class, multiplied by the boosting weights and finally by another factor that will take into account the difference between the two predictions to weight the error based on the distance between the two classes, this because as mentioned several times, in this case exists an ordinal relation between different ages and so it is right to take this aspect into account during errors evaluation.

### 3.2 Regressor loss function

Regarding the loss function design for the regressor part in the ensemble, we have been inspired by the AAR function. This is useful for many motives: the first is that for sure the loss that we define is in line with the function that evaluate the performances of our system, so minimizing our loss we maximize also AAR function, and on the other hand the AAR takes into account just the factors that we want to minimize, so a variational term called “sigma” and “mMAE” that represents the mean MAE between age groups.

With: $AAR = \max(0; 5 - mMAE=) + \max(0; 5 - \sigma)$

$$\sigma = \sqrt{\frac{\sum_{j=1}^{8}(MAE^j - MAE)^2}{8}}$$

$$mMAE = \frac{\sum_{j=1}^{8} MAE^j}{8}$$

So, we choose to minimize the sum of these two terms (loss = sigma + mMAE).
Ideally, this loss can be seen as a chain of MAEjs, were the algorithm try to minimize the specific MAEjs without neglect the connection between each one, so if one MAEj go down and the other remain in their position, the sigma contribute give more importance to the higher MAEj so the next time the higher ones are that the algorithm try to minimize (not interested to the quantity of samples that belongs to each age groups) and so on.

# 4 Experimental results

## 4.1 Experimental framework

This chapter will express all the choices made for the final model, and a brief rationale will follow for each of them.

The first evaluation made, as usual, was on the available dataset. This one is extremely unbalanced (topic discussed in the appropriate section); we decided to make a split of 74/26 respectively for Training and Validation Set, which very similar to 75/25 that turns out to be widely used when dealing with datasets of a certain size.

Specifically:

- the split was done in the training of each component keeping always the same seed, so as to obtain the same split during each one of the training sessions carried out (otherwise we would have risked using data used for the validation of a part of the model as data to fit another part and vice versa, largely underestimating a possible Test error).

- the data were divided into the two sub-sets in such a way as both to keep all 81 possible ages represented in both and to respect the distribution of the Data Set delivered to us (which should still reflect that of the eventual Test Set).

Next, we reflected on how to evaluate performance, in particular the data provided to us were fully split into train-validation, so about testing, since we were not constrained by the specifications, we decided to use an external dataset ("UTKFace").

The motivations for which we didn't further split the dataset obtaining also a test set were:

- the possibility of using more data for the "training phase" limiting the risk of overfitting as much as possible;

- secondly, we evaluated that, however keeping the true distribution of the data on the Validation Set, the error on this was still a reliable enough parameter to evaluate a possible error on the Test Set.

Obtaining our "best model" took a long time, this is because many choices were made. From specifications we had the possibility of using an ensemble of models to do prediction, giving us several possibilities, but also greatly increasing the possible designable architectures, thus the design choices:

- Certainly, the first one in this sense that we had to face with was on the backbone, in fact, we had to figure out which one to choose, especially since we were intent on something that was pretrained, with weights as much adequate as possible to our tasks. We decided in particular to choose a backbone that was towed on a face recognition task, in fact, in several papers we had found a certain affinity between the two challenges (obviously we could not choose pre-drawn models on age estimation). By developing the application on PyTorch we realized that we did not have many models to choose from. Among the ones we found, we selected InceptionResnetV1 both because it was towed on "VGGFace2" (one of the largest datasets for faces that exist), and also because the network had a not excessively large size, offering very good capabilities with respect to inference time (28,348,424 million parameters).

- Next we selected the actual architecture (explained above), after analyzing several possibilities. A very difficult choice was how to combine the outputs of the two classifiers in boosting in order to achieve optimal performance. After several empirical tests we chose to estimate the ages predicted by the two classifiers via

argmax and respectively create two arrays of 81 values set to zero, except by placing one for the corresponding element at the estimated age and the four previous and subsequent years. This was done to provide as input to the final regressor "an extremely reliable opinion overall" from both classifiers. Indeed, we checked the OR between the two classifiers estimates and the AND to verify their complementarity.

Several other possibilities were evaluated, such as directly giving the output of the two classifiers as the input of the regressor, or first estimating the age through the expected value by directly providing it with the estimated age, or with different decision rules. Among all of these possibilities we verified (empirically) that the one adopted was the choice that maximize performance.

The last parameters estimated by finetuning were the choice of optimizer, scheduler, momentum, batch size, number of trainable layers. In particular, we used:

- optimizer: SGD with learing rate 1e-3 and momentum 0.9

- scheduler: StepLR with step size 10 and range 1e-4

- batch size: 256 this value allows us to evaluate parameters such as "mMAE" and "Sigma" with a sufficient number of elements, ensuring that in each batch will be samples representatives of all ages.

- batch balancing was not done but also tested with different shades of balancing.

- number of trainable layers: dependent on the training session done, however at most by the block "mixed_6a" onwards (22,917,969 of parameters).

-class weights: starting from the estimate of error on different class, flattening through a function (take the flattening coefficients).

### 4.2 Results

The results obtained from the network are:
**Best train loss: 2.878109**
**Best val loss: 3.182525 (SIGMA=1,195172 mMAE=1,987354)**

If we want to approximate the loss on validation to that on test, the AAR obtained would result: 10-3.182525 = 6.817475, (Obviously this is what we are interested in, since it turns out to be the parameter for evaluating performance).

However, we know that the model was chosen on the basis of the Validation Set, so, even if keeping the same distribution and samples difficulty, the test error should still be higher. Certainly, the precautions taken in constructing the Validation Set ensure under certain conditions of similarity with the Test Set that the results are comparable.

Obviously, the sigma, however, affects these results and, in fact, in spite of the different countermeasures undertaken to balance the error, is something that still turns out to be significant. Also because, the imbalance between the number of samples for different ages in the dataset is very high, and this cannot be eliminated, at least not completely. The obtained mMAE value attests how the prediction error among the different age groups is relatively low on average (even compared to human standards), although it turns out to be higher for the outer groups (the sigma value says this). The results still seem to us to be satisfactory although certainly further improvable (impossible to find the best possible model).

### 4.3 Repeatability and stability of the results

In order to ensure that the obtained values could be repeated, also for performance comparation between different models, we use a fixed seed to perform the split between train and validation. To guarantee the stability of the obtained results, we also tried different seed to split, and the performances remain comparable. This gives us the certainty that the results obtained on the validation set are not random, therefore dictated by a more or less “fortunate” split.

### 4.4 Prediction of the results on the test

About testing, since we were not constrained by the specifications, we decided to use an external dataset ("UTKFace"), limiting the ages to that on which our system has been training. The performance obtained with this Test Set are reported below:

| AAR |
| --- |
| 1.543778 |

This result is quite good considering that this Set have totally another distribution of the samples with also different samples: in terms of resolution, size, ethnicity distribution, etc. so not similar surely to the Test Set that will be used for our model. Our intent was just to evaluate in a really difficult situation the performance and eventually the limits of this model. This is the distribution of UTKFace dataset:


However, keeping the true distribution of the data on the Validation Set, the error on this was still an enough reliable parameter to evaluate a possible error on the Test Set.

# 5 Supplementary Analysis

This section will review many of the experimentations we did before arriving at the final mode. Before analyzing the models delivered, many of them were also trained on a different backbone. Initially, a resNet152 was selected with pre-training on "imageNet" (famous dataset for object detection), a backbone later modified since it was less adherent on the task of "age estimation" from the faces.

## 5.1 First architecture:

### Single Regressor

The first model that was analyzed was a single regressor, specifically with resNet152, where we set the number of trainable parameters approximately to 14 million. On resNet152 the input was 224x224, the idea behind was both to evaluate the performance of the model and to realize any improvements made with more complex architectures.

Examples are given below (with resnet152):

- MODEL SAVED IN**: SGDSimpleRegressorModel_lr_03**

We did a training of the backbone with a single regressor without anything else, pulling 80% of the backbone. Among different configurations of optimizer and learning rate (sgd, adamw, rmsprop) we chose the best one i.e. sgd with learning rate at 1e-3 and made it run for 20 epochs. Results obtained:

| Best Train Loss | Best Val Loss |
|---|---|
| 4.85834 | 5.591666 |

- MODEL SAVED IN: **SGDSimpleRegressorModel_lr_03_balanced_0_075**

We did a training of the backbone with a single regressor and nothing else, pulling 80% of the backbone. The same as before but with different sampler aimed at slightly balancing the batches especially with 0.075 as the flattening coefficient:

| Best Train Loss | Best Val Loss |
|---|---|
| 4.23274 | 6.879636 |

Decay by putting **flattening coefficient 0.11**:

| Best Train Loss | Best Val Loss |
|---|---|
| 4.32866 | 7.077482 |

### Single Classifier

Another model that has been analyzed is a single Classifier, specifically with resNet152, where we set the number of trainable parameters was approximately to 14 million. On resNet152 the input was 224x224.

Regarding the classifiers, we conducted Age Group estimation (from ten years to ten

years) as our goal, since we had decided to introduce them as components in a more complex architecture. The performances evaluated in this task for a single classifier were:

| Train Loss | Train Accuracy | Val Loss | Val Accuracy |
|---|---|---|---|
| 0.68 | 69.36 | 0.63 | 72.38 |

Later given the poor performance, to perform this task more adequately, we decided to implement an ensemble done in a way that the second classifier uses classes that overlap with that of the first one. This was a solution to the problem related to age group classifications for errors made on “sample edge” that are very similar to each other but belong to different classes even if they have only one year of difference. So, the second classifier uses age classes of ten year except for the first and the last that goes from 1-15 years old and from 65-81 years old. The results for the second classifier were:

| Train Loss | Train Accuracy | Val Loss | Val Accuracy |
|---|---|---|---|
| 0.53 | 73.14 | 0.76 | 72.383 |

as a decision rule to increase the accuracy on this task we decided to examine the two classifications by evaluating the classifier with higher confidence as the main age group, and exploit "the second one" by stretching the “main group” by 5 years toward the second realized classification. Our performances on all Set.

| without Augumentation | with Augumentation |
|---|---|
| 0.927451 | 0.915440 |

**Complete architecture**

As more complex structure we did a training of the backbone with three cloned heads from layer 4 to the end, one attached to the regressor, the other two attached to two classifiers for age group estimation with complementary groups. As decision rule that maximized group estimation performance, after training these two classifiers, the resulting output was given as input to the regressor along with the feature vector extracted from its part of the backbone. The regressor was trained with the trained and blocked classifiers attached, moreover the backbone of the regressor had a pre-train done with a simple model backbone regressor. Finally at the last training we did two runs of 10 epochs.

Performance on validation test:

**Best val loss: 4.179003**

| Best Val Loss |
|---|
| 4.179003 |

### 5.2 Final architecture:

The second architecture takes its cue from the first but tries to improve as much as possible the discriminant information content provided by the two classifiers to the final regressor, to perform the age estimation task. In fact, we had realized how the range groups of ages we were providing to the regressor were too large, not really supporting the regressor in its choice. So, we decided to replace those two age group estimation classifiers with two age estimation classifiers. These will perform this task in boosting base, so with the second classifier weighing the wrong samples from the first one more heavily, so as to be as complementary as possible. Finally, we introduced a different backbone, the InceptionResnetV1 which has the weights trained on a Face Recognition task on the "VGGFace2" dataset (a task much more similar than that performed by resnet152). At the expense of an otherwise smaller network as a backbone, however, we obtained a more solid base on which to perform our training. Even applying the same training procedure of which applied on the old architecture that was abandoned, there was a considerable improvement. The performance obtained from the two classifiers shows as the classifier became more able to classify the range of the people's age. From the softmax probability given from the two classifiers we try two approaches: and so, obtaining the predicted age or with expected values or with argmax. We test different combinations to understand the best decision rule. Specifically, the results explicated below show the performance of the classifiers in AND and OR on a variable range (1 to 4 years):

Classifiers trained on expected value and evaluated on the argMax and EV pair (that is the best configuration found for these models):

| | AND | | OR | |
|---|---|---|---|---|
| | **Train** | **Val** | **Train** | **Val** |
| **Range +-1** | 30.30 | 32.68 | 70.08 | 73.47 |
| **Range +-2** | 64.35 | 67.79 | 86.91 | 89.30 |
| **Range +-3** | 83.14 | 85.95 | 93.97 | 95.36 |
| **Range +-4** | 91.86 | 93.48 | 96.98 | 97.66 |

Classifiers trained on argMax and evaluated both on argMax (the choice of the delivered model):

| | AND | | OR | |
|---|---|---|---|---|
| | **Train** | **Val** | **Train** | **Val** |
| **Range +-1** | 48.61 | 51.94 | 71.78 | 75.28 |
| **Range +-2** | 74.44 | 77.60 | 87.69 | 90.09 |
| **Range +-3** | 87.28 | 89.48 | 94.27 | 95.56 |
| **Range +-4** | 93.21 | 94.52 | 97.00 | 97.68 |

After analyzing the results reported above, we have decided how to mix these values to give discriminant information to the regressor part (final head).

Another experienced approach, additional to these reported above, was to try to give to

the regressor all the features extracted before SoftMax, after computing some calculation on these. After some tests we decided to not follow this approach.

Below are the performances of other different tested approaches, some of which were used in the delivered model, others rejected.

Regressor with InceptionResnetV1 with two training once long (from mixed_6a) and once medium (from mixed_7a) without batch balancing (chosen for the delivered model):

| Best Val Loss |
| --- |
| 3.545217 |

Regressor with InceptionResnetV1 with two training once long (from mixed_6a) and once medium (from mixed_7a) with a lightweight batch balancing:

| Best Val Loss |
| --- |
| 3.609302 |

Classifiers trained with ArgMax and Personalized Categorical (chosen for the delivered model):

First:

| Best Val Acc |
| --- |
| 52.927051 |

Second:

| Best Val Acc |
| --- |
| 52.853481 |

Classifiers trained with Expected Values and Personalized Categorical:

First:

| Best Val Acc |
| --- |
| 36.855514 |

Second:

| Best Val Acc |
| --- |
| 35.986061 |

Total model with regressor not Balanced, classifiers trained on Expected Values:

| Best Val Loss |
| --- |
| 3.465905 |

Total model with regressor Balanced, classifiers trained on Expected Values:

| Best Val Loss |
| --- |
| 4.870654 |

Total model with regressor not Balanced, classifiers trained on ArgMax (chosen for the delivered model):

| Best Val Loss |
| --- |
| 3,182525 |

### 5.3 Problems encountered and analyzed:

- Age group estimation even with supporting ensemble for age estimation (due to range too large)
- Classifiers with argMax more effective than with expected value to support any ensembles (empirically on tests done)
- Advantages of having a regressor (because of the way in which it is constructed, in problems where we have neighborhood relations to be taken into account in the error, it performs better, respect to a classifier)
- Difficulty in falling below a certain loss threshold regardless of ensemble (difficulty in finding something highly discriminating for age estimation, thus an extremely challenging problem)