Deep Learning Project

Labeling and Classification on Celeba

Filippo Orazi 0000928971
Andrea Rossolini 0000954735

February 2021

Contents

1 Introduction **3**

2 CelebA dataset **3**
2.1 Data gathering 3
2.2 The data 4
2.3 Data preprocessing 5

3 CNN4 **6**
3.1 Architecture 6
3.2 Orientation 7
3.3 Light source 8
3.4 Further developments 8

4 VGG19 from scratch **9**
4.1 Architecture 9
4.2 Orientation 10
4.3 Light source 11

5 VGG19 variation **12**
5.1 Architecture 12
5.2 Orientation 13
5.3 Light source 13
5.4 Further developments 14

6 ResNet-50 **14**
6.1 Architecture 14
6.2 Orientation 15
6.3 Light source 16

7 Residual Block Architecture **17**
7.1 Architecture 17
7.2 Orientation 18
7.3 Light source 19
7.4 Further developments 19

8 InceptionV3 **20**
8.1 Architecture 20
8.2 Orientation 24
8.3 Light source 25
8.4 Further developments 25

9 Conclusions **26**

1 Introduction

This report aims to describe the studies and efforts we carried out to build different convolutional neural network models able to achieve two different classifications: indeed, using pictures of humans, these networks are supposed to recognize, whether the face of the picture subject is oriented to the left, right, or center of the photo. Moreover, we developed models that can detect where the light source comes from in a picture of a human face.

2 CelebA dataset

The data used for this comes from the CelebA dataset, a large scale face attributes dataset with more than 200k celebrity images, each with 40 attribute annotations. The images present a large variation of poses and backgrounds, often with occlusions [1]. It includes

- 10177 identities
- 292588 images
- 5 landmark locations
- 40 binary attributes annotation per image

The objective of this research was to develop and train a neural network that could identify the orientation of the face and the light source of each image. Given that a supervised training requires training and validation sets to have the explicit attribute we had to manually identify those since they are not present in the attribute list of the dataset.

2.1 Data gathering

To speed up the process we developed a python based software application to build a dataset where each of the processed image correspond to its attribute. This application also allowed us to parallelize the work as it was easy to merge those dataset once they were built [2]. In total, we were able to manually classify 10000 images tagged with the light source and face orientation. Then we merged our results with the datasets that other groups manually labelled, reaching a total of 11251 images for the face orientation and 14939 for the light source. Considering that the other group did not label the images in sequence (but in random order), some of the images have been labelled two times.
On these, we ran an inspection to find how similar the two sets were. We ended up with an 84% similarity between the orientation sets and a 52% similarity between the light source sets. Analyzing the differences, we understood that the majority of images that had different labels in the sets were because of the different standard taken into consideration while labelling. Indeed, we were stricter in assigning a "central" label in both groups (light and orientation),

whereas the other groups had more loose labelling. This, ended up in a lot of mismatching that was then manually corrected by us.

	Our DF	Others' DF	Overlapping	Mismatched	Total
Light	10000	10000	4228	2025	14939
Orientation	10000	10194	8110	1279	11251

Table 1: Summary of the datasets and their merge

The images have been divided into two groups with a 70-30 partition. The first group, the bigger one, has been used as the training set for the various models. While the second has been used as the validation set. All the specific number of images in the various settings can be found in the summary table 2. After the training, the various models have been compared based on the results of the validation set. Then, the best two neural nets have been chosen to classify the rest of the images.

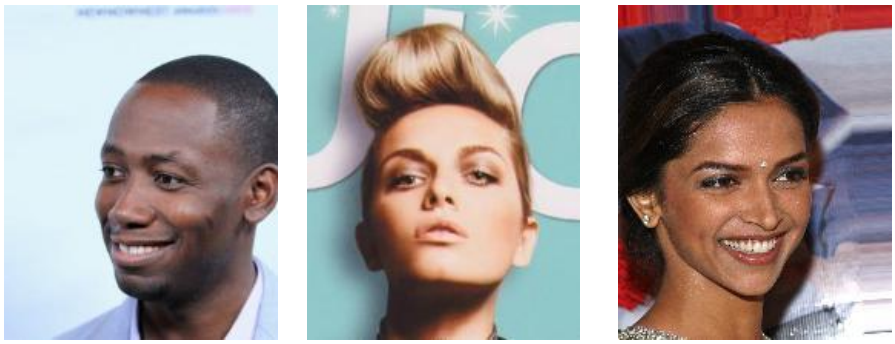
	Training	Validation	Total
Light	10457	4482	14939
Orientation	7876	3375	11251

Table 2: Summary of the sets

2.2 The data

From this dataset (as well as in real life) sometimes it is not possible to say if a source of light came from right, center, or left. For example, in the figure on the right, the whole face is enlightened, but the nose projects a shadow to the left, while the chin projects its shadow on the right. Besides, sometimes the light came from behind the subject and it is difficult to guess the direction of the source. Moreover, sometimes a face is in such a direction that to say whether it is facing one direction or the center is in both cases correct and wrong at the same time.

An example of face orientation can be seen on the figures 1a, 1b, 1c. While, in the pictures 1d, 1e, 1f there are three example for the source of light.



(a) Face to the left (b) Face to the center (c) Face to the right

(d) Light from left (e) Light from center (f) Light from right

Figure 1: Face orientation (a, b, c) and source of light (d, e, f) of different images

2.3 Data preprocessing

Following the good practices of deep learning for computer vision we tried some traditional computer vision preprocessing operations on the images before feeding them to the various neural networks. We tried the following methods:

- Contrast enhancing (Fig: 2c): the contrast of the images was changed with different factors.
- Blur (Fig: 2b) the images were blurred with a Gaussian 5×5 filter .
- Binarization (Fig: 2d) : the images were transformed from RGB to gray-scale and then binarized with respect to their mean .
- Resize: the images were resized with different height and length values, mainly 128×128 and 64×64.

(a) Original (b) Blur (c) Contrast change (d) Binarization

Figure 2: Various preprocessing operation on the same image

We experimented with different combination but we found out that preprocessing didn't have any beneficial effect. In the final trainings we limited the preprocessing to the resize action as it was essential to avoid computational problems.

3 CNN4

The first neural network we tried is the CNN4, a simple convolutional neural network with four convolutional layer presented by Ashesh Chattopadhyay Pedram Hassanzadeh and Saba Pasha [3].

3.1 Architecture

The feature extractor is composed by four convolutional layer with (5×5) filters with increasing depth of 8, 16, 32 and 64; zero padding is used to maintain the size of the input before and after the filters' application. Each convolutional layer is followed by an activation layer with ReLU and the last two ReLUs are them self followed by a MaxPooling layer.
The output of the feature extractor is a $(16 \times 16 \times 64)$ tensor that is flattened and fed to a fully connected layer with 1024 neurons. The final output consists in the the input's probability of belonging to one of the three classes.
The categorical cross entropy is used to compute the loss with the Adam optimizer. Dropout regularization with hyperparameter $p = 0.2$ is used in the fully connected layer to avoid overfitting.

Layer (type)	Output Shape	Param #
img (InputLayer)	(None, 64, 64, 3)	0
conv2d (Conv2D)	(None, 64, 64, 8)	608
activation (Activation)	(None, 64, 64, 8)	0
conv2d (Conv2D)	(None, 64, 64, 16)	3216
activation (Activation)	(None, 64, 64, 16)	0
conv2d (Conv2D)	(None, 64, 64, 32)	12832
activation (Activation)	(None, 64, 64, 32)	0
max_pooling2d (MaxPooling2D)	(None, 32, 32, 32)	0
conv2d (Conv2D)	(None, 32, 32, 64)	51264
activation (Activation)	(None, 32, 32, 64)	0
max_pooling2d (MaxPooling2D)	(None, 16, 16, 64)	0
flatten (Flatten)	(None, 16384)	0
dense (Dense)	(None, 1024)	16778240
dropout (Dropout)	(None, 1024)	0
dense (Dense)	(None, 3)	3075
Total params: 16,849,235 Trainable params: 16,849,235 Non-trainable params: 0		

Table 3: CNN4's structure and number of parameters

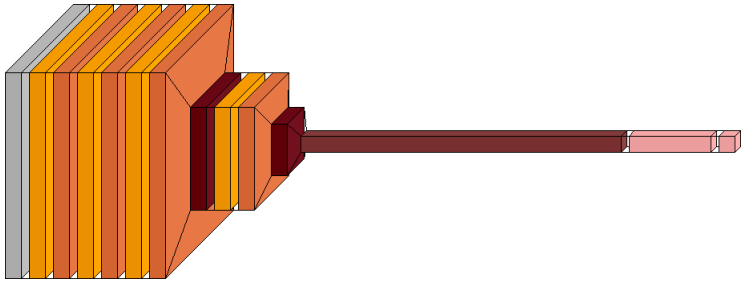


Figure 3: CNN4 visualization

3.2 Orientation

The network was trained for 50 epochs and exposed the tendency to overfit after the first 20 with a rapid increase of the validation loss and the decrease of the training loss.
A fresh model was then trained for 20 epochs reaching the peak of its accuracy while the loss was still acceptable as shown in figure 4.

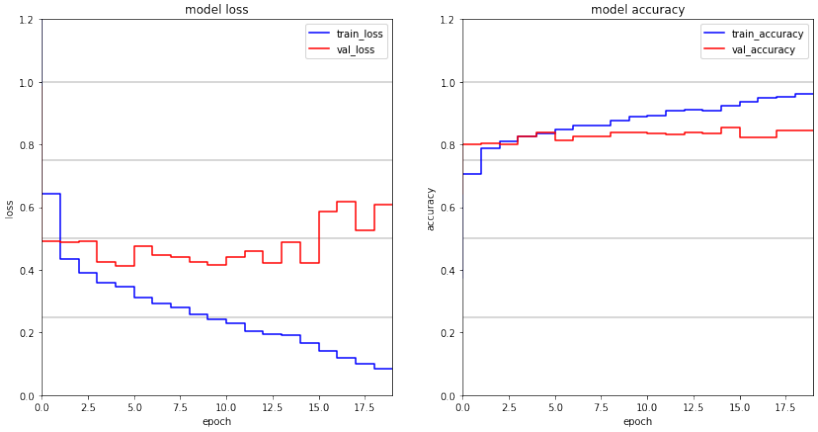


Figure 4: CNN4 Training and validation results for the orientation problem

After the second training the network obtained an accuracy of 84% with a 0.6 loss.

3.3 Light source

The network was trained for 50 epoch and showed signs of overfitting. It was then trained again using an early stop policy that ended after 20 epochs, reaching a accuracy of 84% with a 0.6 loss as shown in figure 5.

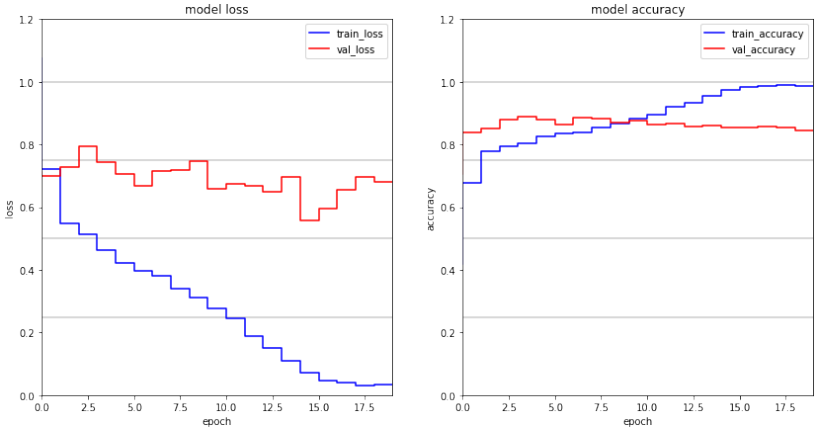


Figure 5: CNN4 training and validation results on the light problem

3.4 Further developments

In the future some action could be taken to improve the outcome of this network:

- Change the p value. Changing the dropout usually allows better results in the long run, avoiding overfitting and leading the network to a more even result between the training and validation set.
- Using a larger, more detailed input. The size of the input largely depends on the computational power available. In our case an input image $64 \times 64 \times 3$ was the best choice to limit the training time of the neural network.

4 VGG19 from scratch

VGG19 comes as a successor of the famous AlexNet. VGG19 is a known deep CNN used to classify images. It carries and uses some ideas from it's predecessors VGGnets, and improves on them. It uses deep Convolutional neural layers to improve accuracy. To train this network we used images 128×128.

4.1 Architecture

The VGG19 architecture is characterized by its pyramidal shape, bottom layers which are closer to the image are wide, whereas the apex layers are deep. The original VGG19 consists of 16 convolutional layers (with 5 pooling layers) and 3 fully-connected layers. For time reasons we removed 6 convolutional layers 3×3 512 and 2 conv layers (256) realizing the architecture shown in table 4. Moreover, given the characteristics of our data set, we decided to add dropout layers, between the fully connected layers at the end, with the purpose of reduce the *overfitting*.

Figure 6: Vgg19 from scratch's structure visualization

Layer (type)	Output Shape	Param #
conv2d(Conv2D)	(None, 126, 126, 64)	1792
conv2d_1(Conv2D)	(None, 124, 124, 64)	36928
max_pooling2d(MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2(Conv2D)	(None,60,60,128)	73856
conv2d_3(Conv2D)	(None,58,58,128)	147584
max_pooling2d_1(MaxPooling2D)	(None,29,29,128)	0
conv2d_4(Conv2D)	(None,27,27,256)	295168
conv2d_5(Conv2D)	(None,25,25,256)	590080
max_pooling2d_2(MaxPooling2D)	(None,12,12,256)	0
conv2d_6(Conv2D)	(None,10,10,512)	1180160
conv2d_7(Conv2D)	(None,8,8,512)	2401289
max_pooling2d_3(MaxPooling2)	(None,4,4,512)	0
flatten(Flatten)	(None,8336)	0
dense(Dense)	(None,4096)	34148352
dropout(Dropout)	(None,4096)	0
dense_1(Dense)	(None,4096)	16781312
dropout_1(Dropout)	(None,4096)	0
dense_2(Dense)	(None,3)	12291
Total params: 55,668,812 Trainable params: 55,668,812 Non-trainable params: 0		

Table 4: VGG19's structure and number of parameters

4.2 Orientation

The model has been trained using early stops. The condition for the stop were 5 epochs in a row, of non-growing validation accuracy and 3 epochs in a row where the validation loss does not decrease. Studying the graphs from figure 7 we can see that the model reaches a good accuracy in a couple of epochs, despite it has "just" eight convolutional layers.
The final results are 0.87 for the validation accuracy, and 0.36 for the validation loss.

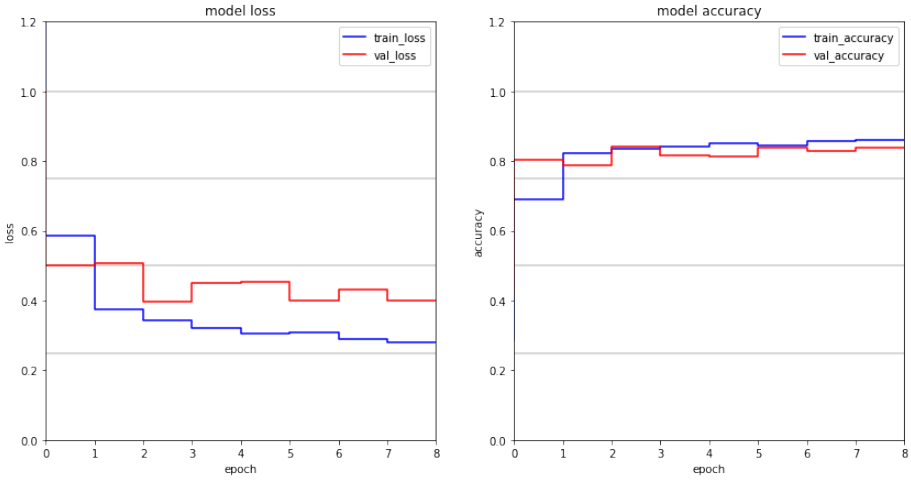


Figure 7: VGG19 from scratch: history on orientation

4.3 Light source

As well for the orientation, the model has been trained to detect the light source. As can be seen from the figure 8, the validation accuracy reaches good values from the beginning, then it stabilizes on a plateau-like. Similarly for the validation loss, it decreases rapidly, and then it stabilizes too around 0.37.
The final results are 0.87 for the validation accuracy, and 0.37 for the validation loss.

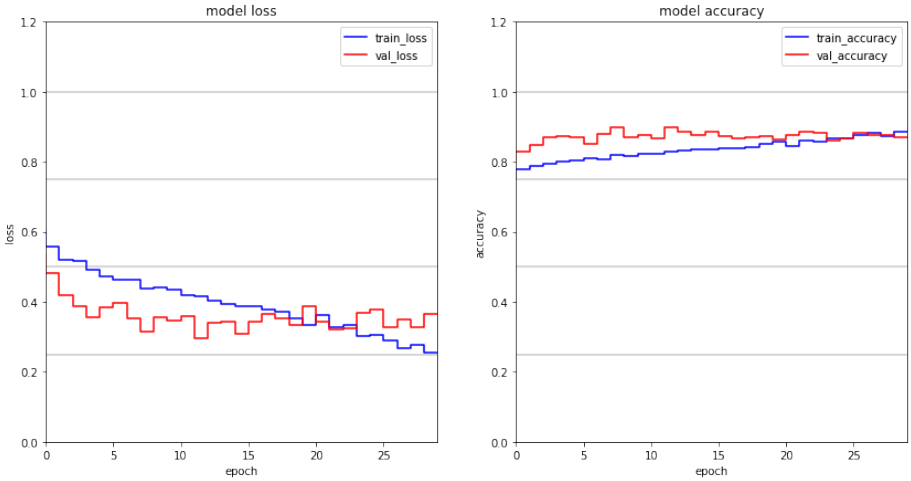


Figure 8: VGG19 from scratch: history on light source

5 VGG19 variation

5.1 Architecture

This neural network is a variation of the classical vgg19, we tried a different version of the convolutional layers to better extract features in the image.
The architecture consists in tree convolutional layers followed by a max pooling repeated three times with increasing depth of channels (32, 64, 128).
The classifier is composed by a hidden fully connected layer.

Layer (type)	Output Shape	Param #
conv2d_14 (Conv2D)	(None, 62, 62, 32)	896
conv2d_15 (Conv2D)	(None, 62, 62, 32)	9248
conv2d_16 (Conv2D)	(None, 62, 62, 32)	9248
max_pooling2d_3 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_17 (Conv2D)	(None, 31, 31, 64)	18496
conv2d_18 (Conv2D)	(None, 31, 31, 64	36928
conv2d_19 (Conv2D)	(None, 31, 31, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(None, 15, 15, 64)	0
conv2d_20 (Conv2D)	(None, 15, 15, 128)	73856
conv2d_21 (Conv2D)	(None, 15, 15, 128)	147584
conv2d_22 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_5 (MaxPooling2D)	(None, 7, 7, 128)	0
dropout_2 (Dropout)	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_2 (Dense)	(None, 128)	802944
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 3)	387
Total params: 1,284,099 Trainable params: 1,284,099 Non-trainable params: 0		

Table 5: Vgg19 variation's structure and number of parameters

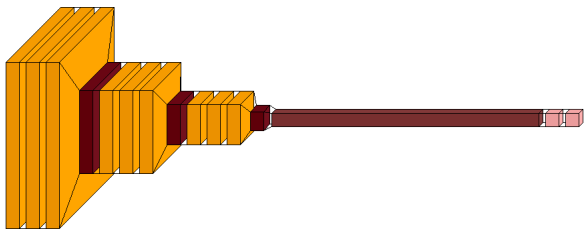


Figure 9: Vgg19 variation's structure visualization

5.2 Orientation

The model was trained on the face orientation problem with the early stop policy and obtained its best performance after 12 epochs with an accuracy of 0.83 and a loss of 0.41 on the validation set

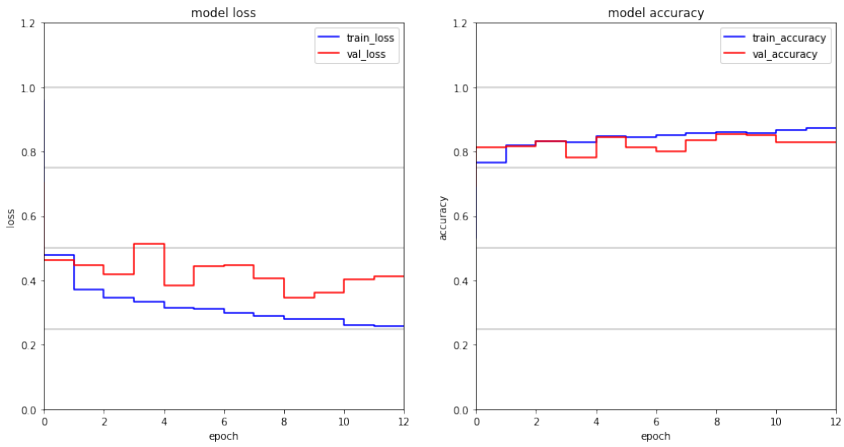


Figure 10: Vgg19 variation training and validation results on the orientation problem

5.3 Light source

The model was trained on the light source with the early stop policy and obtained its best performance after 12 epochs with an accuracy of 0.87 and a loss of 0.34

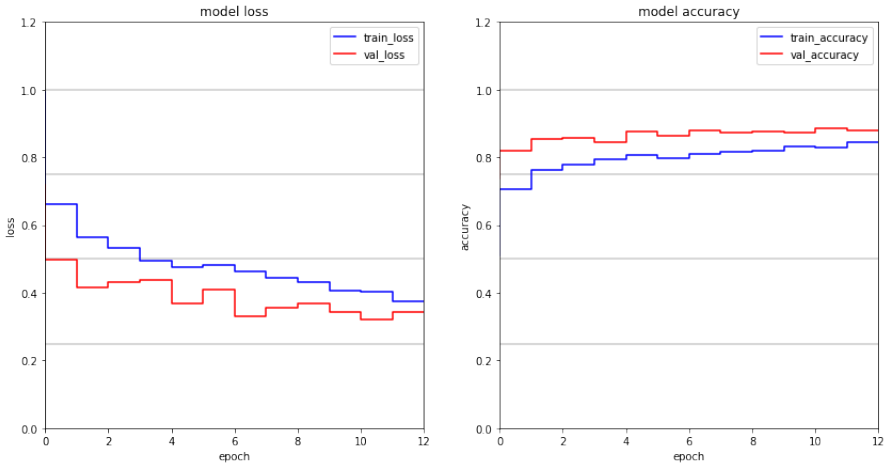


Figure 11: Vgg19 variation training and validation results on the light problem

5.4 Further developments

Comparing the results of the traditional VGG19 with our variation we can assert that the original works better when classifying the face orientation while on the light source the performances are very similar.
That said we should also notice that the small decrease in accuracy for the VGG19 variation comes with the benefit of fifty times less trainable parameters.

6 ResNet-50

The ResNet-50 comes from the work of Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun that presented this architectures to the ILSVRC 2015 classification task competition [4].

6.1 Architecture

It uses recurrent neural network blocks that refer to the input layer instead of learning unreferenced functions, this allows a better training being easier to optimize with respect to other previous architectures. It is called ResNet-50 because it is 50 layers deep, many times more deep than vgg19 but still with a lower complexity thanks to the residual blocks.
The full architecture can be described as in figure 12.

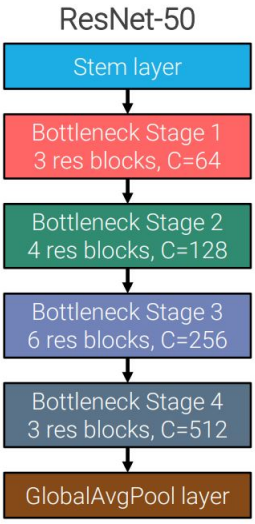


Figure 12: Simplified architecture of ResNet-50

ResNet was studied to overcome the VGG19 approach that required too much computational power to train. They started with a plain baseline inspired

by the VGG19's convolutional layers with 3×3 filters and then followed two simple rules:

1. For the same output feature map size, the layers have the same number of filters.

2. If the feature map size is halved the number of filters is doubled so as to preserve the time complexity per layer.

After the baseline they add shortcut connections which turned the network into its the residual counterpart. The shortcuts are based on simple 1×1 convolutions when there's no change of dimensions [4]. For this particular network called ResNet50 (based on the depth) the shortcuts defines a block called residual bottleneck defined in the presentation paper as in figure 13.

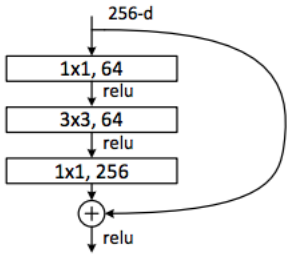


Figure 13: Residual block with no change in dimensions

Table 6 reports the total number of parameters present in the architecture and we can clearly see that the autors were indeed able to diminish the trainable parameters with respect to the VGG19.

Total params:	23,593,859
Trainable params:	23,540,739
Non-trainable params:	53,120

Table 6: ResNet variation's structure and number of parameters

6.2 Orientation

Once the model was trained on the face orientation dataset it was obvious that this architecture was not suitable for the task at hand. The validation loss value spiked various times at different epochs while the training loss kept on descending. The accuracy on the validation set doesn't change through the epochs while on the training set it reaches 98%. This indicates that the solution that the model reached was specific to the training set and couldn't be

generalized.
The full training and validation results can be seen in figure 14.

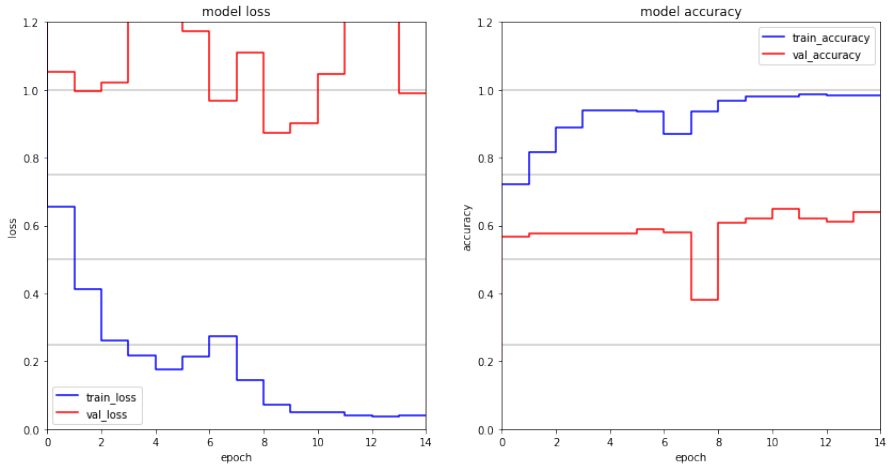


Figure 14: ResNet training and validation results on the orientation problem

6.3 Light source

Training on the light source problem the model behaves very similarly to before as we can see in figure 15. The accuracy obtained on the validation set is 71% and the loss is 0.7. The clear overfit of the model didn't allow it to pass this stage of study and it won't be taken into consideration for the second part, the test on the remaining images.

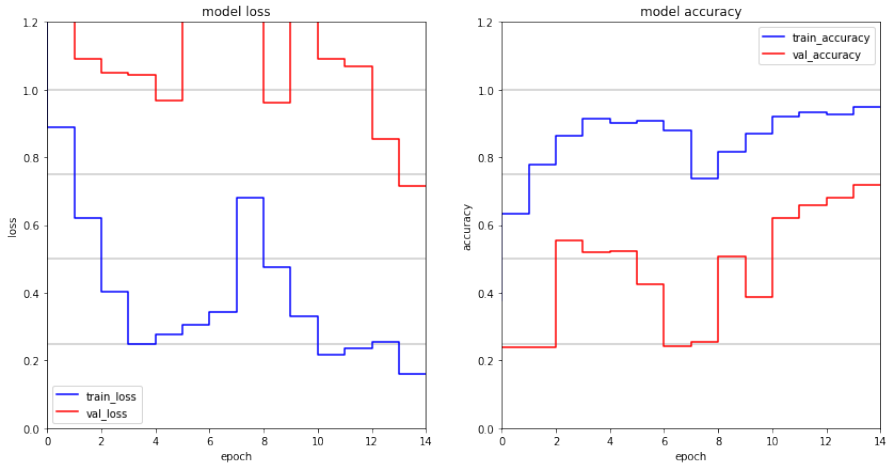


Figure 15: ResNet training and validation results on the light problem

7 Residual Block Architecture

7.1 Architecture

With residual block architecture we refer to a deep learning architecture that we developed based on residual blocks, it consist in 3 residual blocks with (5×5) convolutional filters and an increasing depth of 16, 32 and 64 followed by a fully connected layer of 1024 nodes with 50% dropuout and the output layer.
The residual blocks are made up of two different path, the first one has a convolution, a batch normalization, an activation layer and another couple of convolutional layer and batch normalization. The second path has a convolutional layer with (1×1) filters and a batch normalization. the paths than converge in a sum layer followed by an activation. A representation of the residual block can be seen in figure 16 and the network full architecture can be found in table 7.

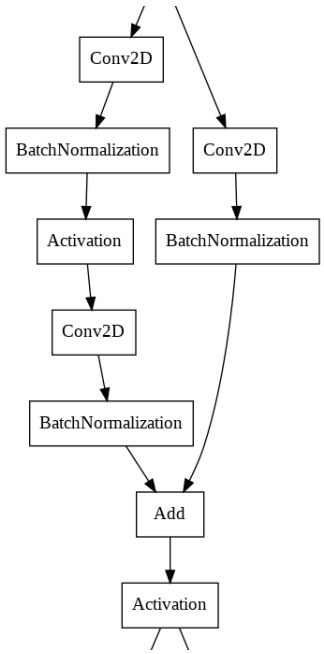


Figure 16: Residual block structure

Layer (type)	Output Shape	Param #
img (InputLayer)	(None, 64, 64, 3)	0
residual block	(None, 32, 32, 16)	7888
residual block	(None, 16, 16, 32)	39392
residual block	(None, 8, 8, 64)	156608
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 1024)	4195328
dropout (Dropout)	(None, 1024)	0
dense (Dense)	(None, 3)	3075
Total params: 4,402,291 Trainable params: 4,401,619 Non-trainable params: 672		

Table 7: Residual block architecture's structure and number of parameters

7.2 Orientation

The residual block architecture found its top performance after 13 epochs of training, up to that point even with the great distance between the losses and the accuracies we see an improvement in the validation results, but this ends around epoch 12 so and goes into overfitting after that. At its best the model has an accuracy of 0.75 and a loss of 0.61. This result is not bad but it doesn't allow the model to candidate as best overall.

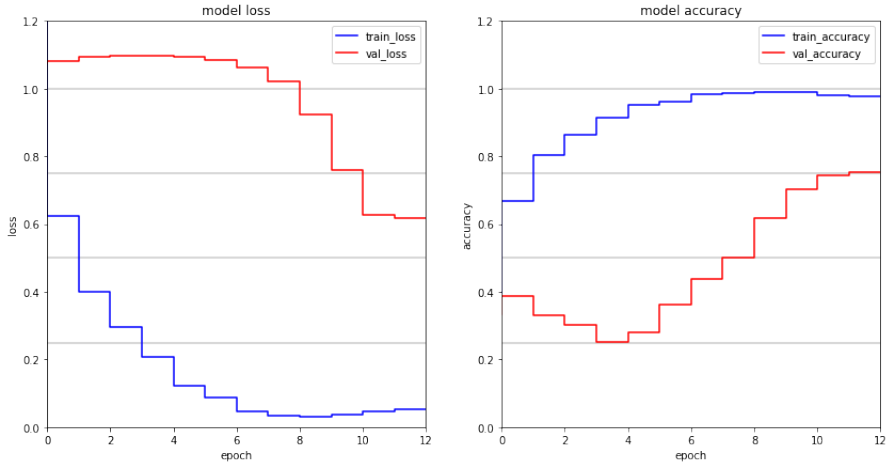


Figure 17: Residual block architecture training and validation results on the orientation problem

7.3 Light source

The model seems to do better on the light source problem where it obtains a final accuracy of 0.83 and a loss of 0.50 in the validation set at epoch 13.

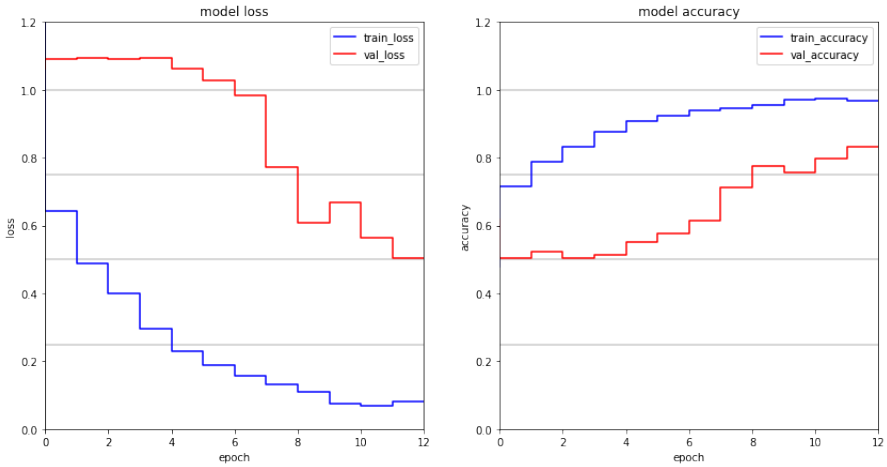


Figure 18: Residual block architecture training and validation results on the light problem

7.4 Further developments

Considering that a dropout layer turn on and off nodes of the network, it is not very used in relation to convolutional layers. Although, following the studies proposed by Sungheon P. and Nojun K. [5], we tried to add dropout layers, with a low rate ($p = 0.1$ or $p = 0.2$) after the convolution networks. This should force the neural network to focus on the more general aspects of the pictures. Thus, the aim of the experiment is to achieve a non-flat accuracy, in other words to "don't stop the learning process". The composition proposed is:

$$Convolution\_layer \rightarrow BatchNormalization \rightarrow ReLu \rightarrow light\_Dropout\_layer$$

The structure of the recurrent block changes as shown in figure 19

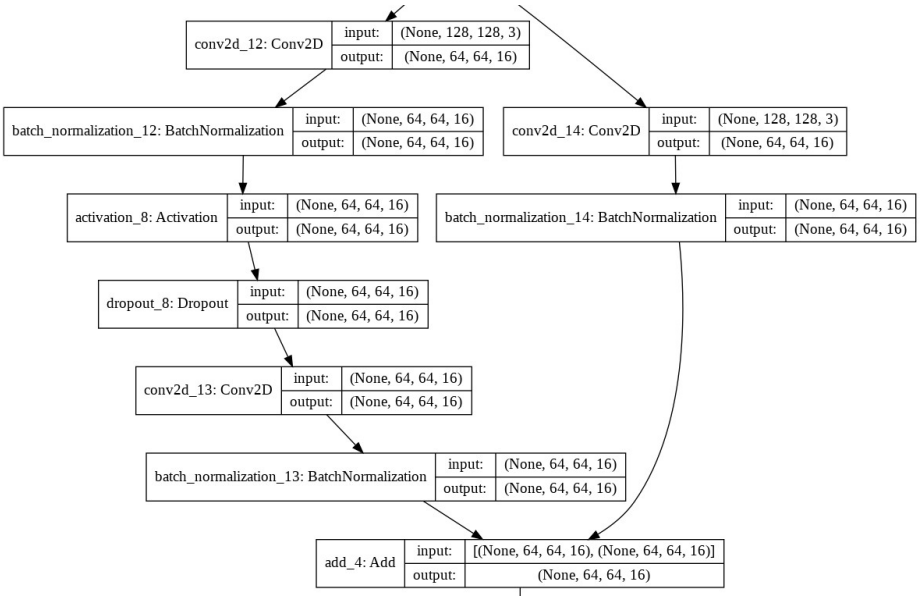


Figure 19: Residual block with dropout

8 InceptionV3

An inception network is a deep neural network with an architectural design that consists of repeating components referred to as Inception modules.
This characteristic allows the network to be more computationally efficient than a network with heavy convolutions as a VGGnet. The Inception V3 model uses different optimization techniques such as factorized convolutions, regularization, parallelized computations, and dimension reductions.

8.1 Architecture

This neural network focuses on consume less computational power starting from the already established inception models. How it is explained in the paper [6], the InceptionV3 is progressively built.

1. ***Inception Module* - Factorization into smaller convolutions**
   This following model is the original inception module (the one used in the *Inception V2 architecture*) and it is not implemented in the final structure. Anyway, it is useful to understand how the other modules are made.

   (a) 1×1 **convolution layer**. The purpose of this convolution is to reduce the dimension of data passing through the network. A convolution operation occurs between the image (input data) and the conv 1x1

filter, to create an output with the dimensions 1 x 1 x n (where 'n' is the number of filters). This cause a reduced amount of channels in comparison with the initial input.Although, a 1x1 filter does not learn any spatial patterns that occur within the image, it does learn patterns across the depth (cross channel) of the image. Therefore it learns the pattern across the channels of an image.

(b) 3×3 and 5×5 **convolution layers**: The purpose of these convolutions is to enable the network to use different convolution filter sizes to learn various spatial patterns across all dimensional components (*height, width, and depth*) of the input (figure 20).

(c) **Max pooling layer**: Pooling downsamples the input data to create a smaller output with a reduced height and width.

(d) **Concatenation layer**: This layer concatenates the outputs of the max pooling layers with the outputs of the convolution layers.

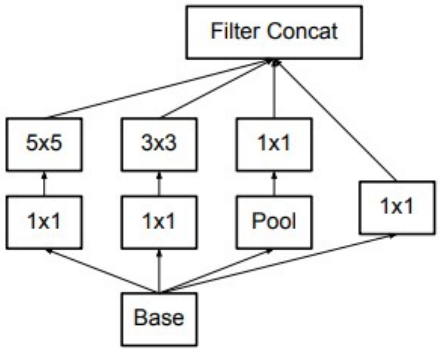


Figure 20: Example of the convolution layers in the inception module

2. ***Asymmetric Inception Module*** **- Spatial Factorization into Asymmetric Convolutions**
This is the main module of the inception V3 architecture. The paper clearly explain that using *asymmetric convolutions* it is possible to obtain better results in terms of computation efficiency. Basically, the paper arguesthat it is possible to replace a $n \times n$ convolution by a $1 \times n$ convolution n followed by a n $\times$ 1 convolution and the computational cost saving increases dramatically as n grows (figure 21). These kind of layers work by taking the x and y axes of the image separately. For example performing a convolution with an $(n \times 1)$ kernel before one with a $(1 \times n)$ kernel.

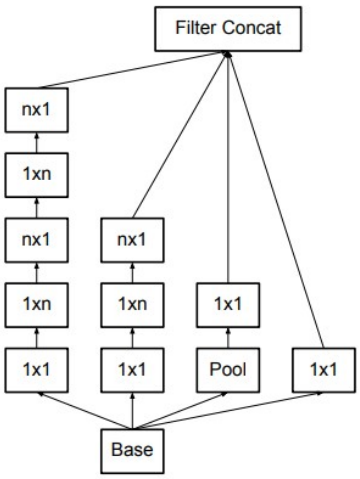


Figure 21: Inception modules after the factorization of the $n \times n$ convolutions

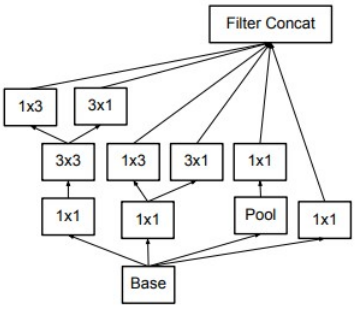


Figure 22: Inception modules with expanded the filter bank outputs

3. ***Auxiliary Classifier Module*** This is a simple CNN inserted between layers during training, and the loss incurred is added to the main network loss. In InceptionV3 the auxiliary classifier acts as a regularizer. The paper argues that it is useful at the end of the training, indeed, networks without auxiliary module reach a slightly higher plateau.

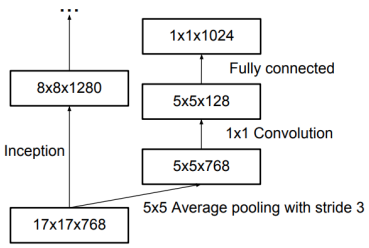


Figure 23: Auxiliary classifier on top of the last layer

4. ***Grid Size Reduction Module***
   Grid size reduction is usually done by pooling operations. However, to combat the bottlenecks of computational cost, a more efficient technique is proposed:

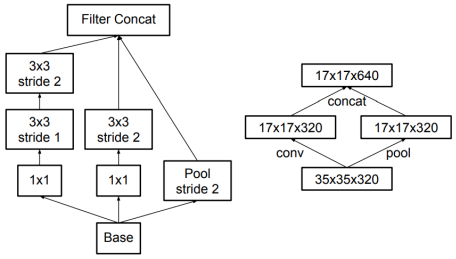


Figure 24: Inception module that reduces the grid-size while expands the filter banks. The diagram on the right represents the same solution but from the perspective of grid sizes rather than the operations.

In figure 25 it is possible to see all the modules explained above in the final architecture. Indeed, we can see that the structure is composed by $\times 3$ *Asymmetric Inception Modules* followed by a *Grid Size Reduction Module*, then other three *Asymmetric Inception Modules*, finally, the *auxiliary classifier*, another *Grid Module* and at the end two *Asymmetric Inception Modules* of the type in figure 22

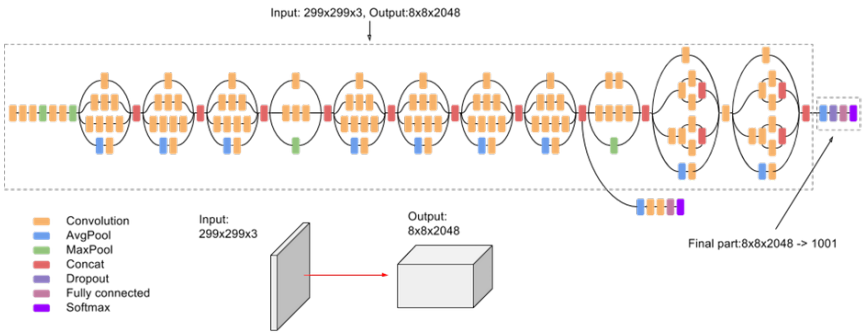


Figure 25: Final architecture from Keras

Total params: 27,118,307
Trainable params: 27,083,875
Non-trainable params: 34,432

Table 8: InceptionV3 parameters

With the aim of reducing the *overfitting* this structure has been enriched with 5 dense layers, each of one is followed by a dropout layer.

8.2 Orientation

From the figure 26 it is clear that the validation loss does sky rocket jumps. This could mean that the model is not converging well to unseen data, meaning that its not seeing a enough similar trends from training data to validation data. Thus, each time the weights are adjusted to better suit the training data, the model becomes less accurate for the validation set. We tried to modify the learning rate, but this just results in overfitting and/or in lower final accuracy. Probably this is due the poor training set wrt the complexity of the network.

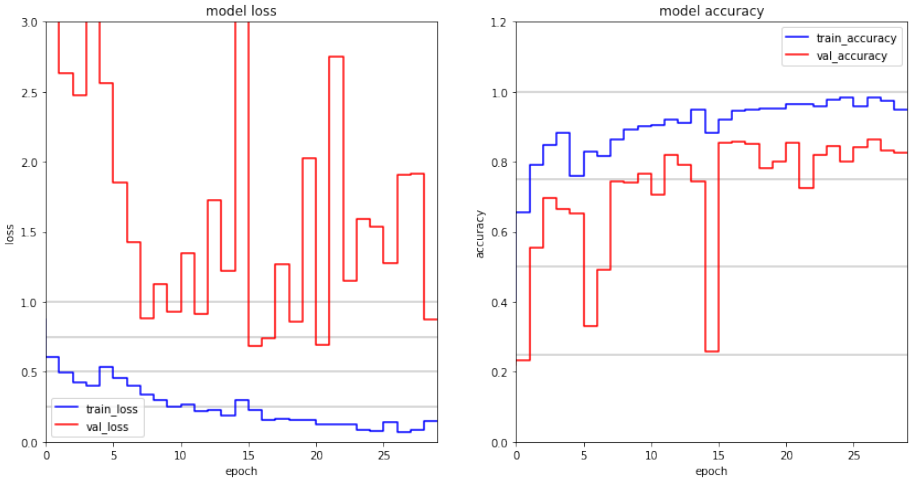


Figure 26: InceptionV3: history on orientation

8.3 Light source

We can make the same consideration we did for the orientation results.

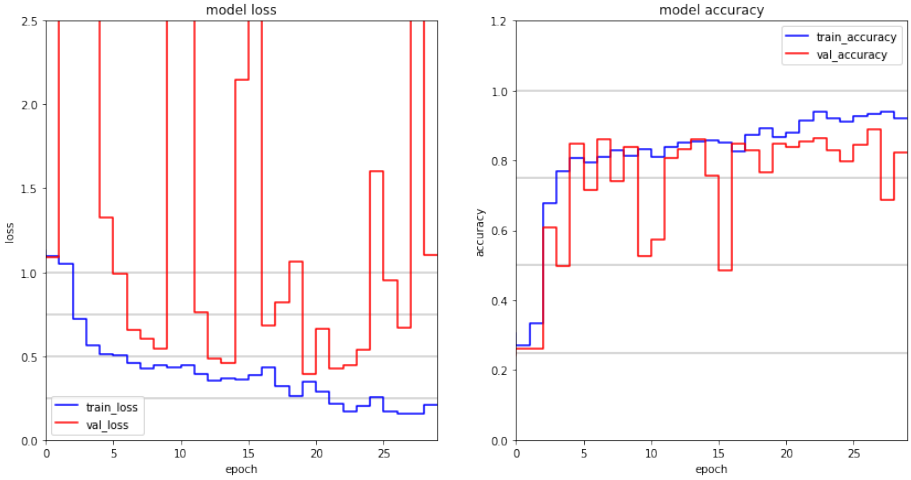


Figure 27: InceptionV3: history on light source

8.4 Further developments

Due to the results we obtained, we decided to not use this model for the labelling all the remaining images. Although, it would be interesting to train this model again but using the all 200k labelled data and study the results.

9 Conclusions

We considered a total of six models to solve the tasks. The best results over the light source problem were the VGG19 from scratch and VGG19 variation hinting that the VGG baseline is a good architecture for this study.
Regarding the orientation problem we have that the best model are the VGG19 from scratch with a 0.87 accuracy and the CNN4 with 0.84 as we can see in the summary in table 9.

Name	light acc.	light loss	orien. acc.	orien. loss
CNN4	0.84	0.68	0.84	0.60
VGG19 *from scratch*	0.87	0.37	0.87	0.36
VGG19 *variation*	0.87	0.34	0.83	0.41
ResNet	0.71	0.71	0.63	0.99
Residual block architecture	0.83	0.50	0.75	0.61
InceptionV3	0.82	1.10	0.83	0.87

Table 9: Models and their performance on the validation set.

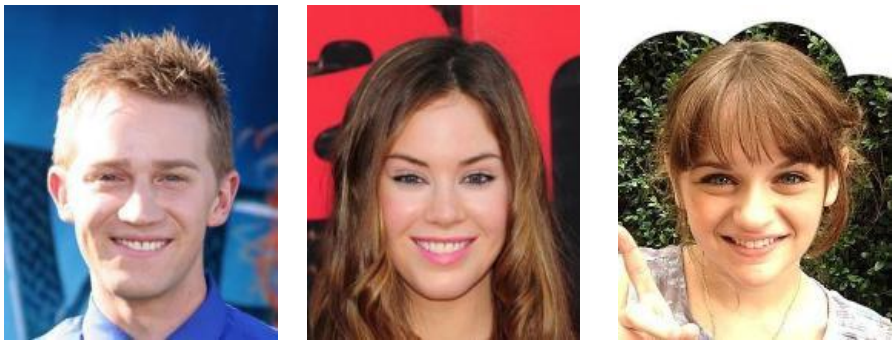
Given these results, we decided to bring to the test two models to find the light source and two to find the position. CNN4 and VGG19 *variation* were chosen to predict the light source, while CNN4 and VGG19 *from scratch* for the orientation. Within the repository, we attached the prediction of each chosen model and a final dataset that contains the merge of the previous with human correction built as explained below.
We compared the prediction of the models and their confidence. This comparison leads to different results in the final labelling of the dataset based on the following conditions:

1. The models outputted the same predictions and at least one of them has high confidence: the image is automatically labelled and stored.

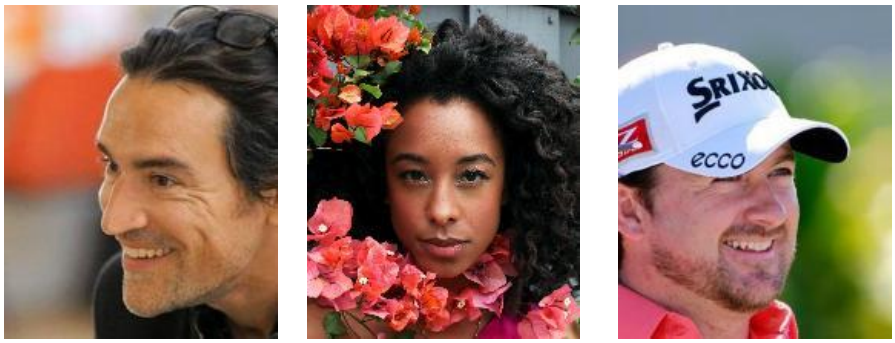
2. The models outputted different predictions but one of them had low confidence while the others were high: the image is labelled with the high confidence prediction. This difference in confidence usually means that one model didn't learn a pattern to recognize this kind of images while the other did.

3. the models outputted predictions with low confidence: the image is not labelled and it will be labelled manually by us as required.

Below we can see an example of automatic labelled images with their confidence.



(a) light_left: 0.9999 (b) light_center: 0.9999 (c) light_right: 0.9998



(d) left_pose: 0.9980 (e) center_pose: 0.9966 (f) right_pose: 0.0.9999

Figure 28: Prediction for light (a, b, c) and orientation (d, e, f) of different images

As a final consideration, we wanted to remark the dataset has a lot of noise and some photo are hard to classify even for a human as no ground truth is available. This could influence the training of the models and ultimately their accuracy.

References

[1] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[2] Andrea Rossolini Orazi Filippo. Celeba classification utils. url-https://github.com/filorazi/CelebaClassificationUtils/tree/main, 2020.

[3] Ashesh Chattopadhyay, Pedram Hassanzadeh, and Saba Pasha. Predicting clustered weather patterns: A test case for applications of convolutional neural networks to spatio-temporal climate data. *Scientific Reports*, 10, 01 2020.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[5] Sungheon Park and Nojun Kwak. Analysis on the dropout effect in convolutional neural networks. pages 189–204, 03 2017.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.