Colorization

Neural Network Course 6cfu

Omar Bayoumi: 1747042

Cristiano Bellucci: 1760390

Graziano Specchi: 1620431

Table of contents

1. Introduction

In this project we implemented, **trained**, **validated**, and **tested** a neural network capable of **colorizing grayscale images**. The model is inspired by the paper: *"Colorful Image Colorization", Richard Zhang, Phillip Isola, Alexei A. Efros*.

The problem has been formulated as a **classification** among classes of colors, computed in the *CIE Lab* space, opposed to the standard ***RGB*** color space.

We implemented a model that involves a pipeline of Convolutional Layers (Convolutional Neural Network, **CNN**), trained over **16K images**. The aim is to take as input a grayscale image and output a colorized version of it. From what the project concerns, there is no need of having output equal to the ground truth, the important thing is that the output of the neural network is **plausible**.

We focused on the aspect of **color saturation**. Indeed, in previous works, the models were not able to predict bright colors, because typically they are orders of magnitudes less frequent than unsaturated ones. Then, following what the paper suggested, we implemented a loss function "ad-hoc", so that, through a rebalancing factor, it avoided the overfitting toward unsaturated colors.

We performed the computation on **Google Colab**, with a limited availability of computational power, RAM, and storage. Then, the results are often not comparable to the ones obtained by the authors of the paper, but still, we obtained some nice realistic results that we will show in the next chapters.

As a metric that we use to check the goodness of the predictions on the test set, is to give the output of our model as input to a pre-trained classification model and check if it classifies them correctly. To do this, we used **VGG-16**. In the end, we are happy about the results we obtained, since, even if they are not always good like the results of the original paper, they suggest that by employing more resources and data, this model could potentially achieve the state-of-the-art.

2. Description and formulation of the problem

Let's consider an example: a grayscale image. At a first glance, we can notice that a lot of information is missing, but if we observe it by paying attention, we can infer which are the colors of the objects. Indeed, the grayscale image still contains shapes, shadows, and other features. For instance, if we recognize that in an image there is the sky, we will color it as blue, or if we recognize a tree, we will color it as a consequence of our object classification. This type of prior knowledge can be learned by a model (in this case, our CNN) and it can be exploited to predict **plausible** (even if different from the ground truth) colors.

Omar Bayoumi: 1747042
Cristiano Bellucci: 1760390
Graziano Specchi: 1620431

We represent images in the **CIE *Lab* colorspace**. The input of the model will then be the solely channel *L* of the image, and the output will be the two channels *A* and *B*.
Then, at training time, the model will learn, for each pixel in the image, the corresponding color class *A* and *B*.
More precisely, we built a 2D plane whose coordinates are the channel *A* and the channel *B*, and we took the part relative to the visible colors, and we divided it in a grid, **so that each texel of the grid corresponds to a class**. Then, there will be a subset of the color space in which all the points will fall into the same texel, namely, into the same class. Of course, the smaller the texels are, the more precise the solution will be. Since the number of texels corresponds to the number of classes, the dimension of the final layer of the network will be the same as the number of texels (the model is a classifier), and smaller texels will require more training time and computational power.
In a **classification problem**, it is important to study some statistics about the training set. If a class is rare in the training set, formally we can say that the training set is **unbalanced**. This will result in a low propension of the model to predict that class. Really bright colors and other kinds of colors are rarer with respect desaturated ones, then, if one uses a loss function that doesn't take care of this fact, of course the model will be **biased towards unsaturated ones**. One could think to apply some "data augmentation" to the training set, but the problem is that is difficult to apply these techniques in this case. Indeed, the paper suggested to let the training set as it is, and act on the loss function itself, giving higher weight to less frequent colors and smaller weight to more frequent colors, to avoid a biased learning of the model.
At the test time, once the problem output the prediction, we still have many choices for the reconstruction of the colored image. We will show an algorithm that uses the concept of **simulated annealing**, by using a temperature parameter T, that makes a significant improvement with respect to a naive one.

3. Model

The structure of the model we built can be seen in *figure 1*. Basically, it consists of a **pipeline of Convolutional Layers**, with different size, stride, padding and dilation. The usage of Convolutional Layers is typical when the input of the model is an image, since convolution is one of the main techniques used to process images also in case the model is not a neural network. **The layers are grouped by 2 or 3, creating 8 sub-convolutional blocks**. At the end of each sub-block there is a **batch normalization layer** that allows every layer of the network to do learning more independently. It is used to normalize the output of each block, so that

learning becomes efficient, and it works as regularization to avoid overfitting of the model. At the end there is an upsample layer and a denormalization function.

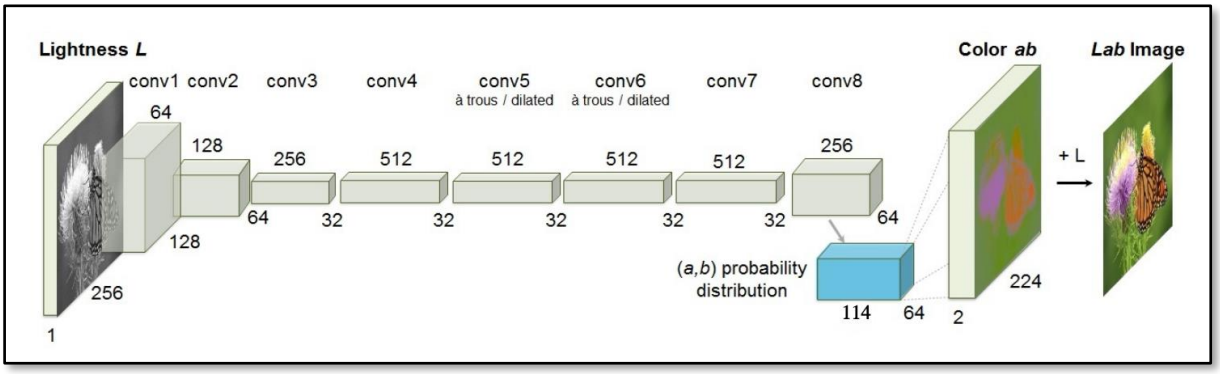


Figure 1. Structure of the model

4. Dataset and Preprocessing

As a dataset, we downloaded the *LSVRC-2015-ImageNet*, and we used 16K images for the training set, 3K images for the validation set and 6K in total for the test set (we used 3K for testing pure grayscale images and 3K for original colored images).

The **pre-processing** is a fundamental part of the whole system. It makes a difference in the learning capability of the model, and it influences the performance of the model a lot. First of all, both at training and testing time each input image is converted into *Lab* space, it gets resized into a 256x256 image that contains (obviously) just one color dimension, i.e., the *L* channel. At training time, for each pixel, the two channels *A* and *B* are used to find the corresponding class among the 114 classes. Then, we build a label tensor that will contain 1 at the position of the right class and 0 in all the other entries. This tensor will be used as a ground truth, to compute the loss function of output and label, that is required for the back-propagation algorithm.

The choice of the *Lab* space is due to the *L* channel representing luminance, i.e., the color intensity that would represent black and white in *RGB*. So, if we can predict the *A* and *B* channels, which represent the color, and reconstruct the colored image by putting them with the original *L* channel, we can reconstruct the original color of the image, or, in any case, we can obtain a realistic colored version of it.

So, the aim is to build classes that identify a specific pair of channels *A* and *B*. First, we take just a slice of the whole color space with channel $L = 50$ (*Figure 2a*). This is graphically represented by a circle in *RGB* where the *x*-coordinate is the *A*-channel, and the *y*-coordinate is the *B*-channel. In this way, given a pair (x, y) it is possible to know the *RGB* color (r, g, b) associated with the color in $Lab\ (50, x, y)$. However, the high number of possible pairs made it necessary to carry out a quantization in a 16x16 grid where each cell is the average of

Omar Bayoumi: 1747042
Cristiano Bellucci: 1760390
Graziano Specchi: 1620431

16x16 adjacent pixels (*Figure 2b*) (obviously the pixels outside the circle did not affect the average).
At this point, **224 classes** have been created. For each pixel, given a prediction in one of these classes, it will then be possible to reconvert by taking the *A* and *B* channels associated with the *RGB* color of the predicted class (or an appropriate calculation, see the *From Class Estimates to Image Reconstruction* section), assign as channel *L* that of the original image and then reconvert to *RGB*.
Subsequently, through an empirical analysis of the frequencies of the classes throughout the training set, it was noted that many classes never appeared and were therefore eliminated, resulting in **114 classes**.

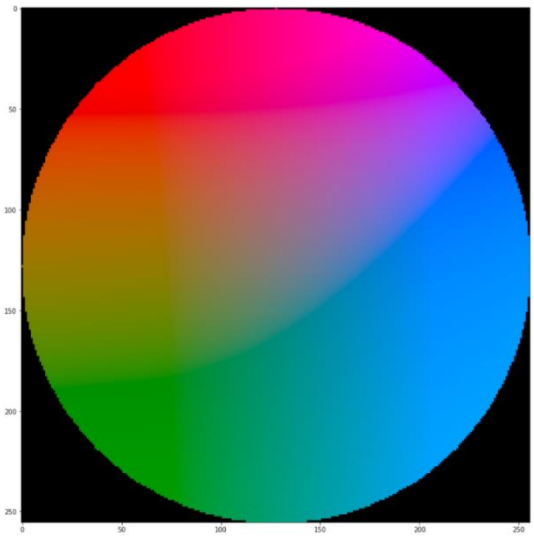


Figure 2a. Lab color circle at L = 50

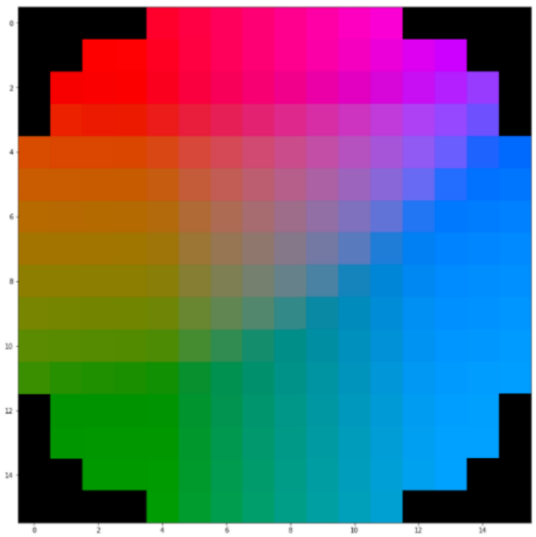


Figure 3b. Color classes

5. Loss Function

Considering that the problem is formulated as a **multinomial classification**, we can now see the details of the loss function. Given an input lightness channel $\mathbf{X} \in \mathbb{R}^{\mathbf{1\times H\times W}}$, the objective is to learn a mapping $\hat{\mathbf{Y}} = \mathbf{F}(\mathbf{X})$ to the two associated color channels $\mathbf{Y} \in \mathbb{R}^{\mathbf{Q\times H\times W}}$, where **H** and **W** are the image size and **Q** is the number of classes. The actual formula is the following (*Formula 1*):

$$\mathrm{L}_{cl}(\hat{\mathbf{Z}}, \mathbf{Z}) = -\sum_{h,w} v(\mathbf{Z}_{h,w}) \sum_{q} \mathbf{Z}_{h,w,q} \log(\hat{\mathbf{Z}}_{h,w,q})$$

Formula 1. Loss Function

Omar Bayoumi: 1747042
Cristiano Bellucci: 1760390
Graziano Specchi: 1620431

The values with the **"hat" (^) are the predicted ones**, while the others come from the ground truth. $\mathbf{Z}_{h,w,q}$ **is the ground truth label** of the pixel at *(h,w)* for the current class *q*. Let's study the formula in detail: there is an external summation and an inner summation that has a **weighting factor**. Besides the weighting factor, **the inner part is a Multinomial Cross Entropy Loss between predicted and ground truth**.
During the experiments, we saw that without a weighting factor, the results were desaturated, and the quality of the predictions were not as good as the case in which we used the weighting term, that we are going to see in detail.
Each pixel is weighed by factor $w \in \mathbb{R}^Q$ (Q is the number of classes), based on its closest *AB* bin. To obtain a **smoothed empirical distribution** $\tilde{p} \in \Delta^Q$, we estimate the **empirical probability** of colors in the quantized *AB* space $p \in \Delta^Q$ from the full training set and **smooth the distribution with a Gaussian kernel** with sigma equal to 5, as suggested in the paper. **We mix the smoothed empirical distribution with a uniform distribution** with weight $\lambda \in [0,1]$ and take the reciprocal as can be seen in *Formula 2*.

$$\mathbf{w} \propto \left((1-\lambda)\tilde{\mathbf{p}} + \frac{\lambda}{Q}\right)^{-1}$$

Formula 2. Mixed Probability

Notice that we used $\lambda = 0.5$ as the paper suggested. Then, we **normalize** so the weighting factor is 1 on expectation (Formula 3).

$$\mathbb{E}[\mathbf{w}] = \sum_q \tilde{\mathbf{p}}_q \mathbf{w}_q = 1$$

Formula 3 - Expectation of w

Since **w** is a **114-dimensional** vector, we take the value that corresponds to the color ground truth class, as we can see in *Formula 4*.

$$v(\mathbf{Z}_{h,w}) = \mathbf{w}_{q^*}, \text{ where } q^* = \arg\max_q \mathbf{Z}_{h,w,q}$$

Formula 4 - Final weighting factor

In this way, for each pixel, there will be a class-dependent weighting term that will avoid the problem of overfitting towards the most-frequent classes.

Omar Bayoumi: 1747042
Cristiano Bellucci: 1760390
Graziano Specchi: 1620431

6. Training and Validation

The model that we are presenting has been trained for 72 hours, in *Google Colab*. We can observe the plot of validation and training loss functions in *Figure 3*.

In the submitted notebook we showed the tests of two different models:

- **The early stopping one, epoch 13**. It was trained up to this epoch because at that point the loss of validation was increasing;
- **The last epoch, epoch 47**.

We noticed that the predictions of the latter are more realistic with respect to the early stopped one. This could be since the loss function tells how much the prediction is **close** to the **original** sample, while we are looking for something that is a bit **relaxed**, namely, even if the result is different from the original, we prefer a more realistic prediction.

We wonder that if we used a larger training set, we could obtain more realistic results and also closer to the original one. The reason behind this, is because the Neural Network overfitted on the small number of cases in the training set, while maybe in the validation set there are classes that the model has never seen before. Then, our hypothesis is that the training set doesn't contain enough data to allow the network to learn generalization enough.

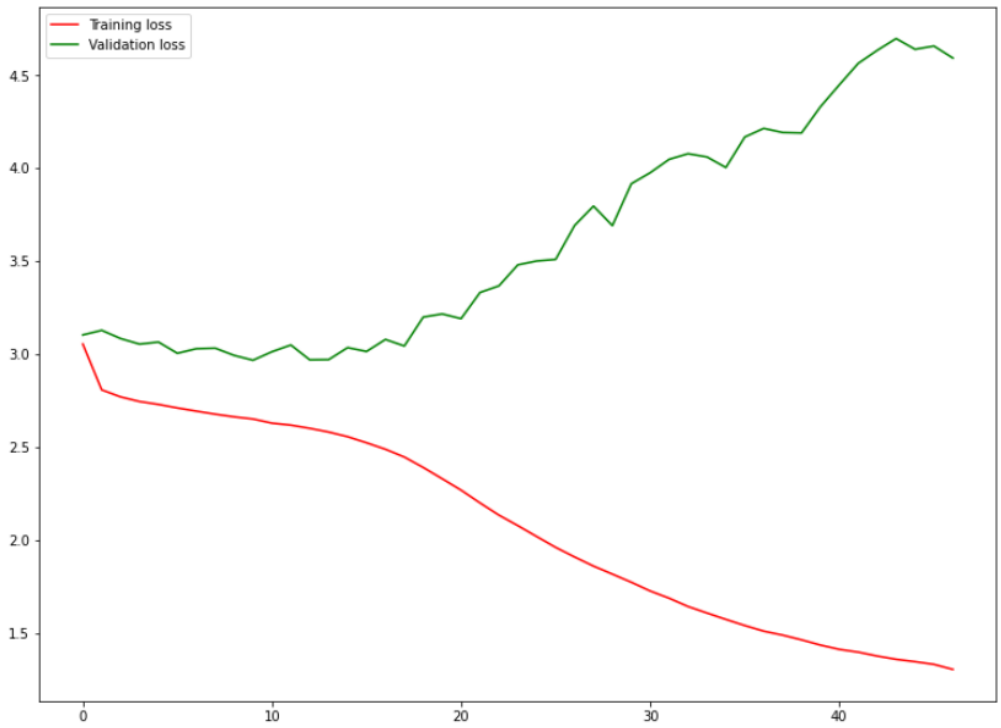


Figure 3. Red: Training loss function. Green: Valid loss function.

Omar Bayoumi: 1747042
Cristiano Bellucci: 1760390
Graziano Specchi: 1620431

7. From Class Estimates to Image Reconstruction

Once we get the predicted distribution for each pixel, we need to map it into a point estimate in *AB* space. One choice could be to take the mode of the predicted distribution for each pixel, in this case the result would be more vibrant but sometimes spatially inconsistent, like spots of uncorrelated colors in a part of the image that should have a uniform color. Another choice could be to take the mean in place of the mode, but this would produce desaturated results, exhibiting an unnatural sepia tone.

Then, in order to get the best from both options, **we interpolate by re-adjusting the temperature T of the softmax distribution and taking the mean of the result**. This is inspired from the *simulated annealing* technique, and thus refer to the operation as taking the **annealed-mean** of the distribution, as we can see in *Formula 5*.

$$\mathcal{H}(\mathbf{Z}_{h,w}) = \mathbb{E}\big[f_T(\mathbf{Z}_{h,w})\big], \quad f_T(\mathbf{z}) = \frac{\exp(\log(\mathbf{z})/T)}{\sum_q \exp(\log(\mathbf{z}_q)/T)}$$

Formula 5. Simulated Annealing

In this formula, H is the function that takes the predicted distribution and f_T is the function that takes as input the output of the network and computes the annealed-mean with a given fixed parameter T. **Setting $\boldsymbol{T = 1}$** leaves the distribution unchanged, lowering the temperature T produces a more strongly peaked distribution, and **setting $\boldsymbol{T \to 0}$ results in a 1-hot encoding at the distribution mode**.

In the Colab Notebook that we submitted, there are several examples in the output of the cells. An example of application of different values for the temperature parameter on the same output image can be seen in the *Figure 4*.

We can notice that with high temperature, the colors get desaturated, while as soon as the temperature gets close to zero, the colors in the image start to become more vibrant, more bright but also less spatially consistent. Indeed, in the case of $T = 0$, the dog has some red spot on the head, and this is of course a weakness of the prediction, because a human would likely recognize that the image is not real.

We noticed that in our case, the value $\boldsymbol{T = 0.09}$ is the one that corresponds to the best result. Also, in the test phase with the **VGG-16 classification**, we will see that the images reconstructed by using that temperature parameter are classified better with more confidence with respect to the other ones.

Omar Bayoumi: 1747042
Cristiano Bellucci: 1760390
Graziano Specchi: 1620431

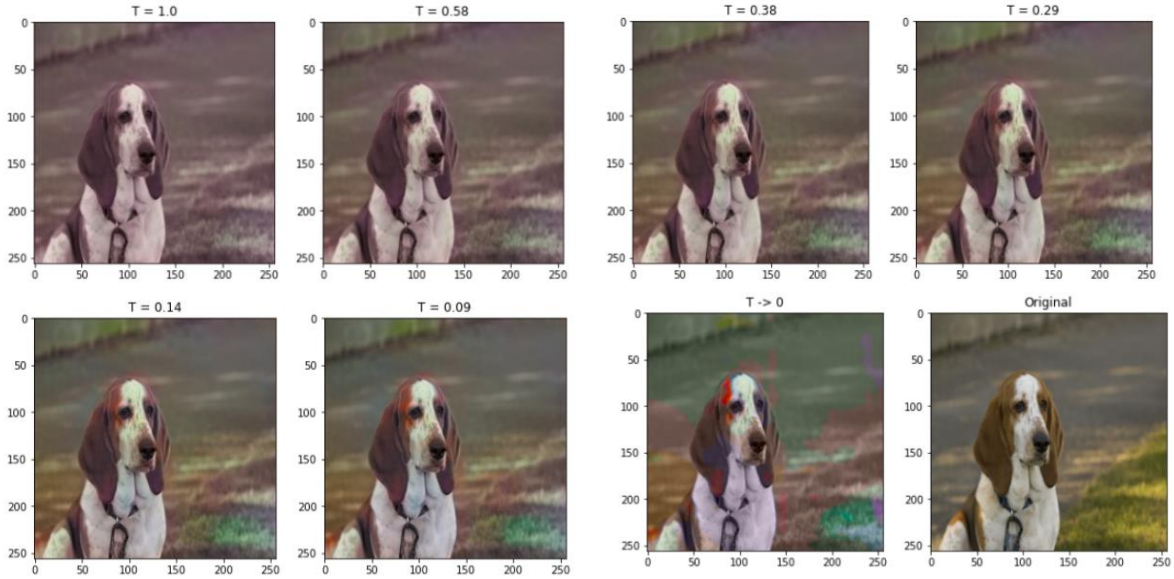


Figure 4. Application of Simulated Annealing and different temperatures T.
Notice that the last image in the lower right corner is the original image

8. Tests

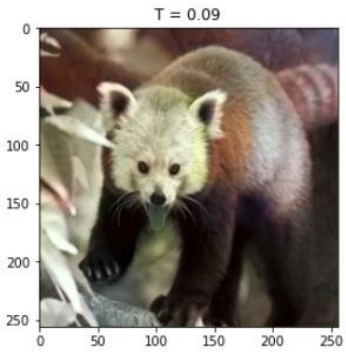
First, a baseline test for the evaluation of the performances of the model can be just looking at the reconstructed images and notice if they are visually coherent with the results that we expect or not. Indeed, in Figure 3 we can see that for $\boldsymbol{T = 0.09}$ the result is quite acceptable, considering also the computational power and time that has been allocated for the training of this model. However, an interesting metric can be to feed a **classification model (VGG-16 in our case)** and see if that model performs good classification. We also compared the predictions of the classification by giving as input the grayscale image one time, and then by giving the reconstructed image the next time. In many cases, the result is that the recolored image is far better classified with respect to the grayscale one. This is also a very interesting point because this means that a lot of information that was missing in the grayscale image, is somehow recovered by applying our model. Let's take for instance the example in *Table 1*. The higher confidence corresponds to $\boldsymbol{T = 0.09}$, meaning that VGG-16 better recognizes the recolored image rather than the original one. Of course, this is a rare case, but still, it is surprising. We can also notice that the classification on the grayscale image is still high, meaning that, even if in that image a lot of information is missing, the VGG-16 network can still exploit and find patterns that are present in the image. With the $\boldsymbol{T \to 0}$ instead, we see the worst classification.
Finally, we show a list of results, and the corresponding classification of VGG-16.

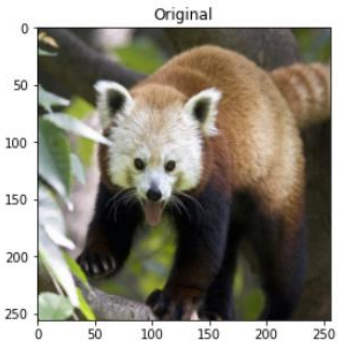
Omar Bayoumi: 1747042
Cristiano Bellucci: 1760390
Graziano Specchi: 1620431

```
ORIGINAL : basset 0.9923388957977295
-------------------------------
 t=0.09 :  basset 0.9959808588027954
-------------------------------
 t=0 :     basset 0.9531656503677368
-------------------------------
 BW IMG : basset 0.976791501045227
```

Table 1. Predictions performed by VGG-16 of the basset in Figure 3.
ORIGINAL: classification on the original image.
T=0.09: classification on the reconstructed image with T=0.09.
T=0: classification on the reconstructed image with T —> 0.
BW IMG: classification on the grayscale image (the one given as input to the model).

Test 1. Lesser Panda



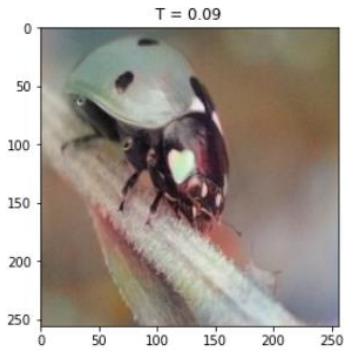


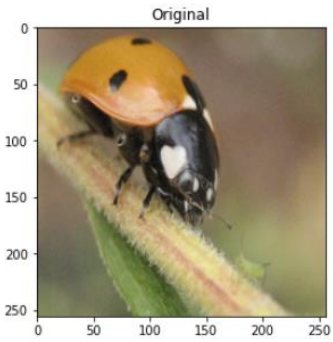

```
-------------------------------
ORIGINAL : lesser panda 0.9968859553337097
-------------------------------
 t=0.09 :  lesser panda 0.9997159838676453
-------------------------------
 t=0 :     lesser panda 0.9953823685646057
-------------------------------
 BW IMG : lesser panda 0.9938421249389648
```

In this example, it is surprisingly to see that the prediction of VGG-16 on the recolored at $\boldsymbol{T = 0.09}$ has more confidence with respect to the original image. Of course, this could be due to the specific classification model we are using, so, for instance if we use another classification model we could have different results. Still, this result could be nice for both visual checking and for the classification model.

Omar Bayoumi: 1747042
Cristiano Bellucci: 1760390
Graziano Specchi: 1620431

Test 2. Ladybug

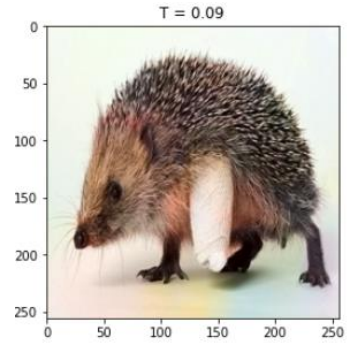


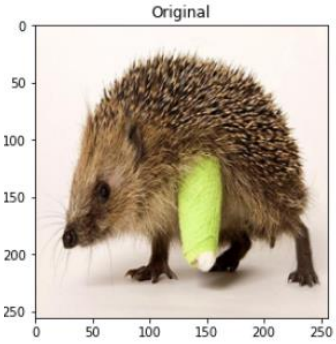



```
------------------------------
ORIGINAL : ladybug 0.7062113881111145
------------------------------
 t=0.09 :  rhinoceros beetle 0.9738121628761292
------------------------------
 t=0 :     rhinoceros beetle 0.2820638120174408
------------------------------
 BW IMG : rhinoceros beetle 0.9938040971755981
```

In this case, we can notice that the colorization completely failed the typical color of a ladybug. Indeed, it colored the ladybug with blue/grey tonality. Also, the predictions have been misguided from this colorization error.

Test 3. Porcupine



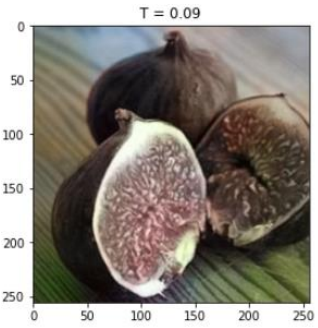


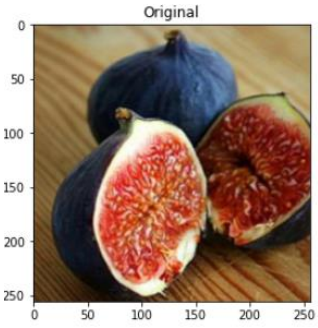

```
------------------------------
ORIGINAL : porcupine 0.9998527765274048
------------------------------
 t=0.09 :  porcupine 0.9997132420539856
------------------------------
 t=0 :     porcupine 0.9871244430541992
------------------------------
 BW IMG : porcupine 0.9996265172958374
```

In this case, we don't have evidence of improvement of image data with respect to the grayscale one, but this could be due to the fact that VGG-16 is "good enough" that it is enough to give the grayscale version of the image in order to have a good classification. Still, we can notice how well the neural network colored the animal, and this is a proof of the fact that VGG-16 alone cannot be used as a valid metric.

Omar Bayoumi: 1747042
Cristiano Bellucci: 1760390
Graziano Specchi: 1620431

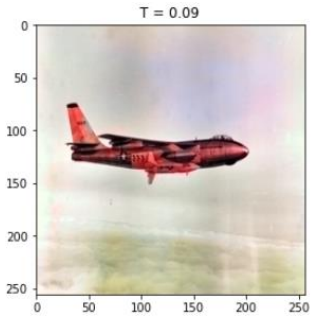
Test 3. Fig

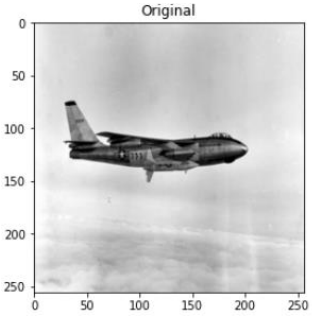





```
------------------------------
ORIGINAL : fig 0.9964054822921753
------------------------------
 t=0.09 :  fig 0.9694618582725525
------------------------------
 t=0 :     fig 0.42716217041015625
------------------------------
 BW IMG : fig 0.5833533406257629
```

Test 4. Warplane






```
------------------------------
ORIGINAL : warplane 0.6147164106369019
------------------------------
 t=0.09 :  airliner 0.36480918526649475
------------------------------
 t=0 :     warplane 0.5198606252670288
```

Omar Bayoumi: 1747042
Cristiano Bellucci: 1760390
Graziano Specchi: 1620431