**University of Porto - Faculty of Engineering**

Master in Informatics and Computing Engineering

**Project Report**

# Colorized Classification

Colorization CNNs as pre-training for the Classification task

Miguel Amorim
`up201907756@edu.fe.up.pt`
Rita Mendes
`up201907877@edu.fe.up.pt`
Sérgio Estêvão
`up201905680@edu.fe.up.pt`

May 2023

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

## Contents

## List of Figures

## 1 Introduction

Colorization is a classic challenge not just in the realm of computer vision, but also in the world of art. One of the most difficult aspects of colorization is constructing extra visual information only from greyscale images. Grey-scale images can translate into numerous alternative colorized images using this additional generating information, which is known as multi-modality, for example, a flower can have distinct colors but the training set only has one definite answer to this. This multi-modality has also been one of the most challenging issues in representative modeling. Colorization and translation are two examples of multi-modality in real-world applications.

Concurrently, pre-training approaches have received a lot of attention as a way to improve neural network performance. Training a network on a large dataset to learn general representations, which can later be fine-tuned for specific tasks using smaller labeled datasets, is what pre-training entails. This transfer learning method has been shown to be successful in overcoming data constraints and minimizing processing and computational requirements.

The main aim of this report is to investigate the potential of leveraging the encoder part of a colorization neural network as a pre-training mechanism for classification tasks, which could improve a neural network's classification skills by using the knowledge gained during colorization, which fundamentally entails comprehending visual content and context.

Considering this, in the first stage, we will show our implementation of colorization neural networks. Following that we will compare and analyze the impact of the transfer learning process on a classification neural network, utilizing the previously created colorization neural network as the pre-trained task.

To close this report, there will be a section for conclusions where we will analyze the obtained results. This will also include a short discussion of how the concepts applied for each optimization can be used in other situations and use cases.

## 2 Colorization

As mentioned by the author in [2], Colorization has been a challenging question in the area of computer vision and machine learning. It is challenging because the gray-scale image corresponds to several colorful images, which means that there is more than one correct answer. There have been several approaches to this task after the neural network is used widely in the field of computer vision.

Regarded as one of the most successful, the authors of [1] approached Colorization as a regression problem. They used the LAB color space, where they converted the input images to the LAB color space, gave the neural network the lighting images (grayscale), and made it train to output the $AB$. This was achieved by classifying the Lab space into 313 ab pairs and applying multinomial cross-entropy loss function to represent the inherent multi-modality in colorization.

The architecture of this neural network, as seen in figure 1, consisted of a deep CNN encoder-decoder structure, the encoder part of the network effectively extracted high-level features from the input grayscale image, capturing its essential visual characteristics. Subsequently, the decoder part of the network reconstructed the colored image based on these extracted features.

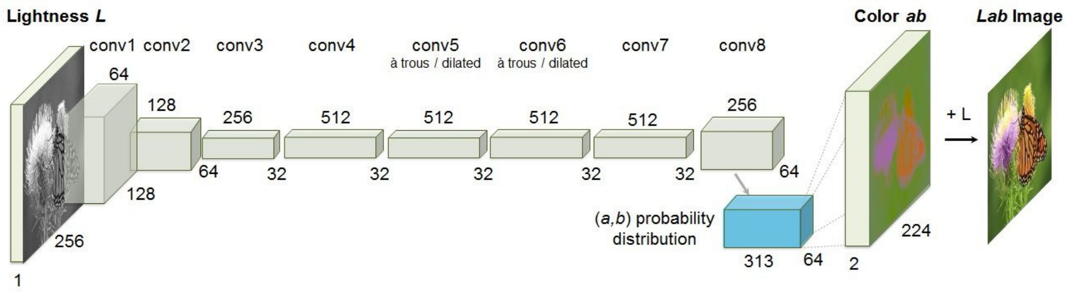

**Figure 1:** CNN architecture from the *Colorful Image Colorization* [2016] paper [1]

## 2.1 Approach

The primary idea behind this model is to take a greyscale image as input, produce colorful images as output by predicting the AB value, then calculate the loss function by comparing the generated colorful images to the original image. Its structure is modified from that of [1], although it also employs Lab scale images and an encoder-decoder architecture. Our approach has less complexity, having fewer convolutional layers and not resorting to classifying the Lab space into separate 313 bins.

The network, as seen in figure 2, consists of consecutive convolutions, each convolutional layer refers to a block to a single convolution, followed by a LeakyReLU layer and, if the convolution changes de tensor dimensions, by a BatchNorm layer. The downscaling of the encoder is performed by normal convolutions while the upscaling is performed by transposed convolutions.

Unlike other approaches, we opted for the LeakyReLU activation function with a 0.2 slope, this was done to handle the "vanishing gradient" problem inherit from the ReLu activation function.

One further detail is that all the convolutions involving dimensionalizing the tensor space have $stride = 2$ and $kernel\_size = 4$, instead of the typical $stride = 1$ and $kernel\_size = 3$ this is to avoid the biasing problem, as mentioned in [3], where specific areas of the image would have a significantly higher impact on the final results due to the convolutions using certain pixels more often.

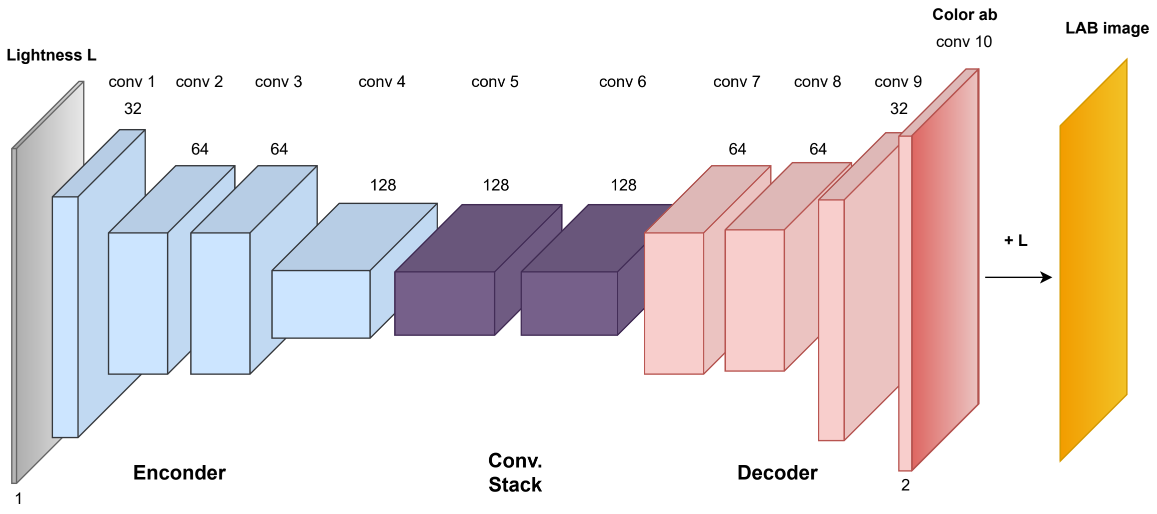

**Figure 2:** Colorization network architecture, it is divided into 3 modules, an encoder, which contains the downsampling part of the model, the conv. stack, a module containing consecutive convolution to draw out features of the image, and the decoder, containing upsampling part of the model

## 2.2 Objective Function

As mentioned in [1], the objective function's job is to, given an input light channel $X \in \mathbb{R}^{H \times W \times 1}$, map a function $\hat{Y} = F(X)$ [1] to the two color channels $Y \in \mathbb{R}^{H \times W \times 2}$, where $H, W$ correspond to the height and width of the image, respectively.

To perform this task we first tried the MSE (mean square error), a criterion that measures the mean squared error (squared L2 norm). The MSE is calculated by taking the average of the squared differences and accounting for errors in both the a and b channels. However, we observed that the MSE criteria were insufficient for our needs. Its consistent consideration of all differences ignores their relative importance. As a result, when an item has a range of different ab values, the best solution based on the Euclidean loss is the mean value of that range. In color prediction, the averaging effect favors grey and desaturated outputs. As a result, the MSE criteria are not suitable for forecasting ab values in this case.

To counteract this issue, we developed a loss function, *WeightedSmoothL1Loss*, which is intended to alleviate the constraints of traditional loss functions in color prediction applications. This loss function improves the Smooth L1 loss by introducing rarity-based weighting to improve ab value prediction accuracy. The loss function computes the Smooth L1 loss between the predicted and target ab values during the forward pass. Histograms of the target ab values are constructed to account for the rarity of different color values, and rarity weights are calculated inversely proportionate to the color counts. These rarity weights are then adjusted so that their total equals one. A color rarity matrix is generated by combining the rarity weights for the a and b channels. Finally, the color rarity is multiplied by the loss function.

The code for the implementation is as follows.

[1]The hat symbol corresponds to a predicted value

```python
class WeightedSmoothL1Loss(nn.Module):
    def __init__(self, weight_factor=1):
        super(WeightedSmoothL1Loss, self).__init__()
        self.weight_factor = weight_factor

    def forward(self, input, target):
        log_loss = nn.SmoothL1Loss()(input, target)
        color_counts_a = torch.histc(target[:, 0, :, :], bins=20, min=-110, max=110)
        color_counts_b = torch.histc(target[:, 1, :, :], bins=20, min=-110, max=110)
        rarity_weights_a = 1 / (color_counts_a + 1e-6)
        rarity_weights_b = 1 / (color_counts_b + 1e-6)
        rarity_weights_a = rarity_weights_a / torch.sum(rarity_weights_a)
        rarity_weights_b = rarity_weights_b / torch.sum(rarity_weights_b)
        color_rarity = rarity_weights_a * rarity_weights_b
        weighted_loss = torch.sum(log_loss * color_rarity)
        weighted_loss *= self.weight_factor
        return weighted_loss
```

## 2.3 Experiments

The experiments were run with 2 distinct loss functions, MSE and the *WeightedSmoothL1Loss*, the Adam optimizer, with a learning rate of $1e^{-3}$ and no weight decay, and the models were trained for 50 epochs.

### 2.3.1 Over-fitting

The main issue of our training was over-fitting, as seen in figures 3 and 4, where the validation loss ended up not reducing a significant amount. Initially, we followed the recommendation of the authors in [4] for dropout and managing over-fit in convolutional neural networks, namely using stochastic dropout after the convolutional layers, however, the model only started to converge slightly later, and no significant improvements to the overall performance where noted. To improve the overfitting, a different dataset could be used, or reducing the number of parameters of the model could be a possible measure.

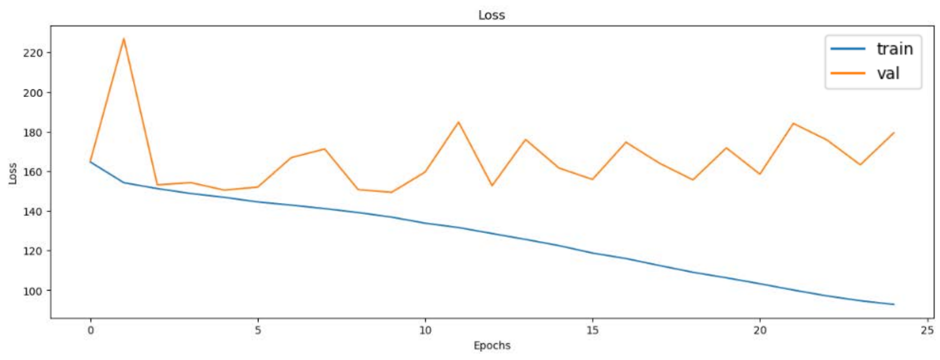

**Figure 3:** Progression of the loss value in the first 25 epochs of training for the model with MSE criterion

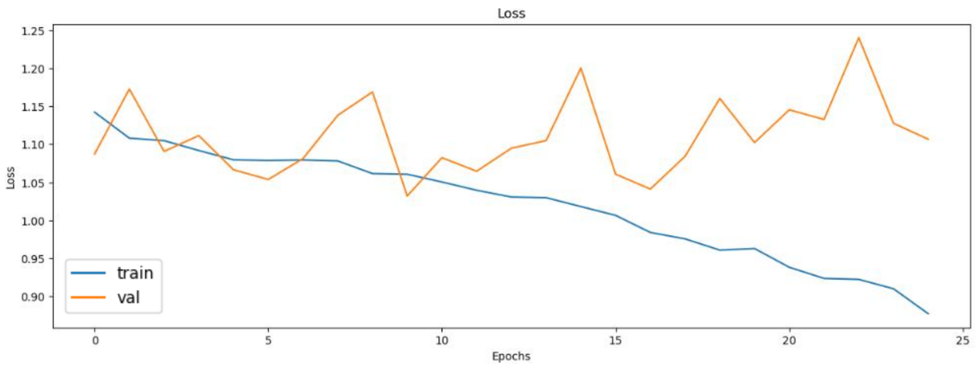

**Figure 4:** Progression of the loss value in the first 25 epochs of training for the model with Weighted Log criterion

### 2.3.2 Results

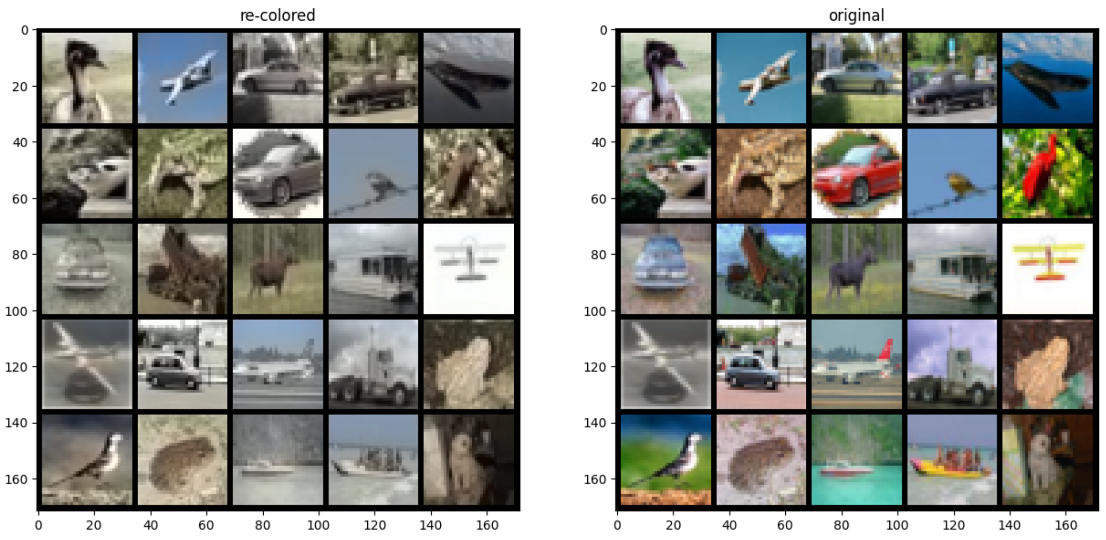

**Figure 5:** Colorization results using the MSE criterion

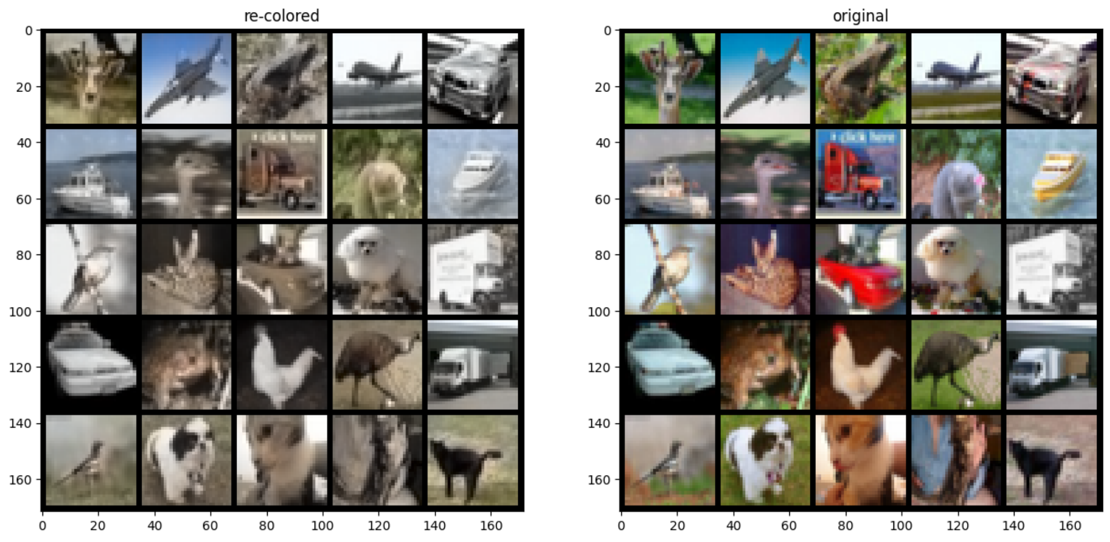

**Figure 6:** Colorization results using the Weighted Log criterion

From figures 5 and 6, we are able to confirm that the colorization using the weighted log criterion ended up generation images slightly better colorized than MSE, having a larger palet of colors displayed, making it possible to say from an empirical standpoint that the Weighted Log outperformed MSE, however, the colorization network's performance fell short of our expectations. Several reasons led to the poor performance.

The colorization task itself posed inherent difficulties. Assigning proper color values to grayscale photographs is a highly subjective procedure that is depending on human judgments. As a result, defining an architecture for the model is a slow process, and the network created struggled to understand the complex links between visual characteristics and related color representations, resulting in inconsistent and inaccurate colorizations.

Furthermore, the colorization network's design and hyperparameters may not have been adequately matched for the unique dataset and job at hand. The network's ability to learn and represent complicated color distributions may have been constrained, limiting its ability to produce realistic and aesthetically appealing colorizations.

However, the results are still promising and we were able to notice improvement and evaluate changes with distinct techniques, architectures, and parameters.

## 3 Colorization as a Pre-task for Classification

In this section, we will explore the use of colorization as a pretext task for image classification. This approach is rooted in transfer learning and revolves around utilizing a model trained for a specific task as the backbone of a model designed for a different one.

### 3.1 Experiments

We will discuss our experiment using the pre-trained encoder of the colorization model as the backbone of a classification model, to see how it compares to a similar randomly initialized encoder.

We structured our neural network into two distinct components: the encoder and the classifier. The encoder consists of the first eleven layers, which mirror the architecture of the colorization model's encoder. On the other hand, the classifier encompasses the remaining nine layers and takes inspiration from the ResNet-18 model, the model's diagram can be seen in Figure 7.

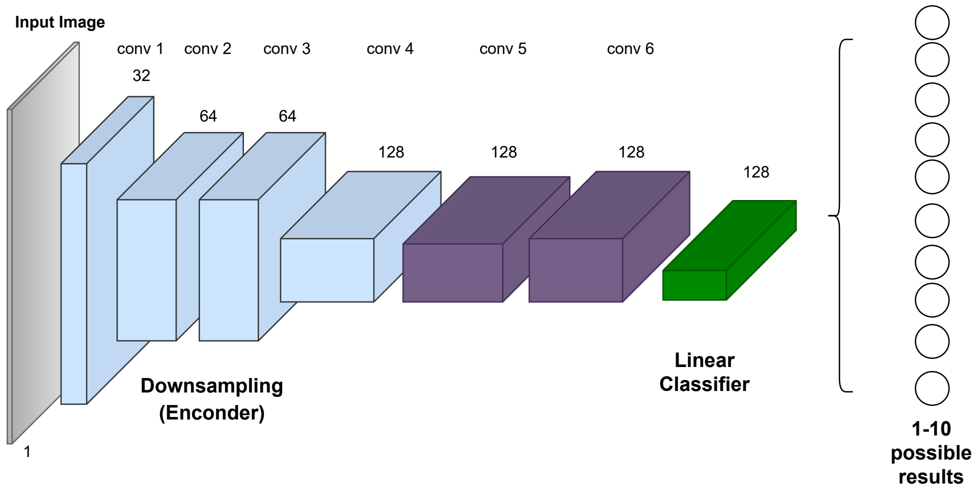

**Figure 7:** Classification network architecture, it is divided into 2 modules, an encoder, which contains the downsampling part of the model, and a module containing the linear classification with its 10 possible results

### 3.1.1 Using the pre-trained encoder

We took the pre-trained encoder, which already had its weights and biases from the colorization task, and utilized it as the encoder for our classification network, observing its behavior training for 25 epochs and comparing it to a similar network whose encoder was randomly initialized.

### 3.1.2 Using a randomly initialized encoder as control

In this section, we have our control recipe, having the same model architecture as the pre-trained encoder but with random initialization of the weights and biases of the model's layers. This was done by sampling values from a random distribution, such as a Gaussian (normal) distribution or a uniform distribution.

By randomly initializing the parameters, you create a model with no prior knowledge or bias about the problem domain. This allows you to compare the performance of other models or techniques against this baseline. If another model or technique outperforms the randomly initialized model, it indicates that the alternative approach has learned meaningful patterns or representations.

Overall, using random initialization as a control in the experiment helps in understanding the relative performance of different models, assessing model capacity, evaluating training procedures, and establishing statistical significance in the results.

## 3.2 Results

To analyze the model's behavior during training, we plot the cross entropy loss and classification accuracy, comparing the model's performance using test and validation data. This comparison is essential to assess the model's ability to generalize and, as a result, detect overfitting.

We hypothesized that the model utilizing the pre-trained encoder would converge in fewer epochs compared to a randomly initialized one. We expected this approach to decrease training time while achieving improved results. However, we also anticipated that the pre-trained model might be more prone to overfitting.

Based on the data presented in Figure 8 and Figure 9, we can observe that the control model exhibits signs of convergence between epochs 10 and 15. In contrast, the experimental model demonstrates convergence around epoch 10, thus confirming our initial hypothesis.

However, the classification accuracy, as depicted in Figures 10 and 11, does not show a significant improvement.

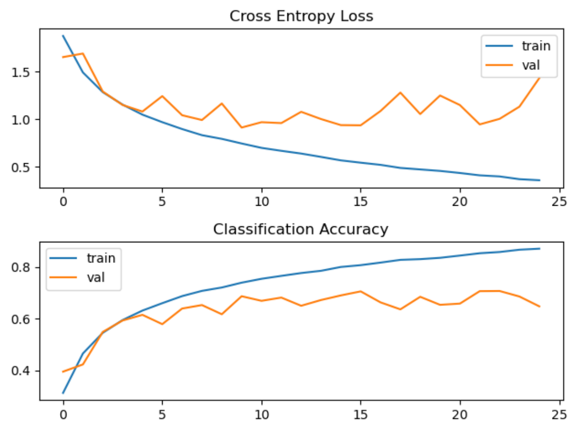

**Figure 8:** Cross entropy loss and classification accuracy during the training of the model with a randomly initialized encoder (control)

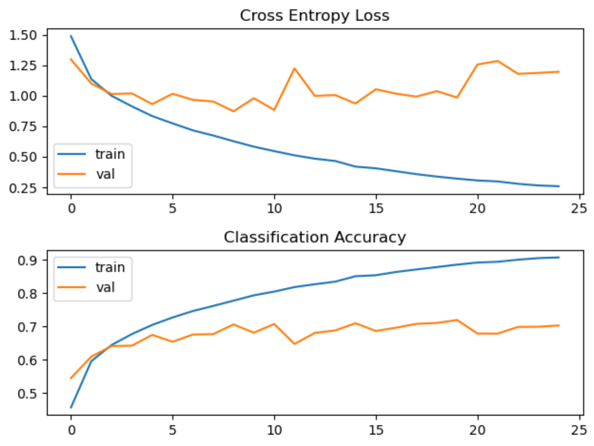

**Figure 9:** Cross entropy loss and classification accuracy during the training of the model with a pre-trained encoder (experiment)

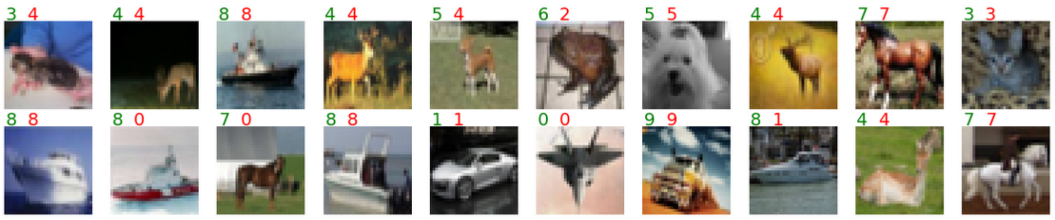

**Figure 10:** Prediction results of the model with a randomly initialized encoder (control)

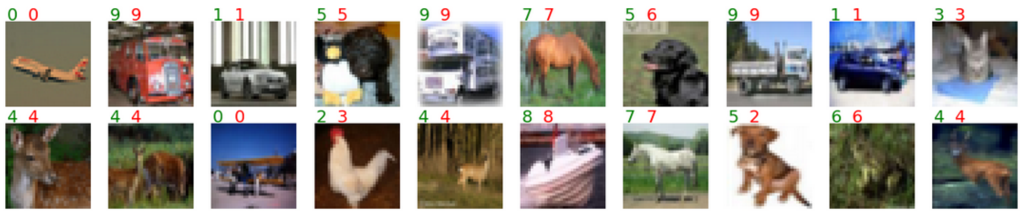

**Figure 11:** Prediction results of the model with a pre-trained encoder (experiment)

## 4 Conclusions

The Colorization CNN has demonstrated potential in the fields of computer vision and image processing. We discovered the colorization neural network's effectiveness in generating colorized pictures from grayscale inputs through experimentation and analysis. Inspired by prior research, the design employs an encoder-decoder structure to extract high-level information and rebuild the colored picture based on these attributes. We investigated other objective functions, such as MSE and WeightedSmoothL1Loss, and discovered that the latter produced more accurate and diversified colorization outputs. However, we encountered overfitting issues, which may necessitate more analysis and possibly changes to the model's architecture or dataset. Nonetheless, the Colorization of CNN's potential as a pre-training strategy for classification tasks proved promising.

While using image Colorization CNN, more specifically the encoder portion of this network, as a pretext task for image classification, we observed through our experiments and analysis that pre-training a classification model with a colorization task can lead to improved classification accuracy, generalization, and accelerating the model's convergence, requiring less training time. The colorization task acted as a form of unsupervised learning, which allowed the model to learn meaningful features and representations from unlabeled data, proving the usefulness of transfer learning in neural network problems.

## References

[1] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," *arXiv preprint arXiv:1603.08511*, 2016.

[2] W. Jang, "Sampling using neural networks for colorizing the grayscale images," *Journal Name*, 2023.

[3] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *arXiv preprint arXiv:1606.04934*, Oct 2016.

[4] S. Park and N. Kwak, "Analysis on the dropout effect in convolutional neural networks," *Graduate School of Convergence Science and Technology*, 2023.