A Project Report

on

Automatic Image Colorization Using Auto-encoders without image compression

to be submitted in partial fulfilling of the requirements for the course on

Artificial Intelligence – CSE 3013

(E2)

By

Lakshit Dua 18BCE0824

Shaik Haseeb Ur Rahman 18BCE0646

Naman Sood 18BCI0168

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Fall Semester 2020-2021

TABLE OF CONTENTS

ABSTRACT

This paper tends to the issue of producing a conceivable hued photo given a grayscale picture. Past ways to deal with tackle this issue have either depended on human inputs in the form of scribbles and hints or resulted in desaturated colorizations. Motivated by [20], we present a completely programmed approach that utilizes profound neural systems to image colorization. We investigate the convolutional neural network space, the optimizers, regularization techniques and learning rate scheduling to comprehend the viable procedures to acquire acceptable colorized images as our output. We train a convolutional neural system and provide an image colourization solution that retains the dimensions of the image. Our model aims to have the same input and output image size. We additionally examine other cutting-edge research done in the field that led us to generating satisfying images with low error rates, including conditional generative adversarial networks [19]. At last, we give a subjective furthermore, visual examination of our outcomes and plots to accompany them, finishing up with roads for future exploration.

Objectives:

- Conversion of black and white images to colored images without human intervention has been the subject of various researches going on within communities of machine learning and computer vision using a unique convolutional neural network approach retaining the image properties and dimensions.
- Our secondary objectives involve improving the efficiency of the model through learning rate schedulers, different optimizers, dropouts and other techniques such as lasso and ridge regression.

Introduction

In the past years, CNNs have emerged as the dominant technology for a host of image related tasks such as classification, labeling, image generation and style transfer, achieving error rates below 5% on the ImageNet challenge. Recently, Convolutional neural network models have been emerging as the state-of-the-art image classifiers and solving problems around it.. These characteristics make it a clear natural choice to explore the auto-colorization task at hand.

They have the ability to discern different patterns, hints and associate them back to the target classes. **We aim to use the robust nature of Convolutional neural networks with image colorization to provide an efficient and accurate colorization of the given input image while retaining the image size** (no form of compression or image property loss). We would be combining this robust property with auto-encoders that are usually used for image classification, data compression and other minor tasks such as image denoising.

This auto-colorization model can further be extended into a variety of different industries. Many large animation companies, sketching and comic book artists use these auto-colorization tools to efficiently color similar panels. This technology can also be incorporated in medical industries where the majority of machines work on black and white sensors (ultra-sound, MRI scans etc.). It can also be used as a preliminary proxy task for data visualization and analysis in data-sets collected from sensors.

REVIEW-1 (Survey & Analysis)

Literature Review

[1] Key concept – Image to image translation

Research – The main objective of the research paper is to understand the mapping required for image-to-image translation. This process is conventionally done using paired training data but in many cases, that data is not available. The authors presented an approach for learning to translate an image from a source domain X to a target domain Y without paired examples. The distribution of X is tuned to be indistinguishable from the distribution Y using adversarial loss functions. The results were then tested upon several image translation tasks such as collection, object transfiguration, season transfer, photo enhancement, etc. Quantitative comparisons against several prior methods demonstrate the superiority of the approach.

Methodology – The authors' approach builds on the "pix2pix" framework of Isola et al, which uses a conditional generative adversarial network to learn a mapping from input to output images. These methods use adversarial networks, with additional terms to enforce the output to be close to the input in a predefined metric space, such as class label space, image pixel space and image feature space. Using the same evaluation datasets and metrics as "pix2pix" the method is compared against several baselines both qualitatively and quantitatively. The tasks include semantic labels↔photo on the Cityscapes dataset, and map↔aerial photo on data scraped from Google Maps. They also perform ablation study on the full loss function.

Conclusion – Although the method can achieve compelling results in many cases, the results are far from uniformly positive. On translation tasks that involve color and texture changes, the method often succeeds. On tasks of dog→cat transfiguration, the learned translation degenerates into making minimal changes to the input. This failure might be caused by the generator architectures which are tailored for good performance on the appearance changes. Handling more varied and extreme transformations, especially geometric changes, is an important problem for future work.

[2] Key concepts – Generative adversarial networks, image translation

Research - The authors investigate conditional adversarial networks as a general-purpose solution to image-to-image translation problems. These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping. This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. They also demonstrate that this approach is effective at synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images, among other tasks. The results in this paper suggest that conditional adversarial networks are a promising approach for many image-to-image translation tasks, especially those involving highly structured graphical outputs. These networks learn a loss adapted to the task and data at hand, which makes them applicable in a wide variety of settings.

Methodology – To explore the generality of conditional GANs, the authors test the method on a variety of tasks and datasets, including both graphics tasks, like photo generation, and vision tasks, like semantic segmentation: Semantic labels↔photo, trained on the Cityscapes dataset [12]. Architectural labels→photo, trained on CMP Facades. Map↔aerial photo, trained on data scraped from Google Maps. BW→color photos. Edges→photo, Sketch→photo: tests edges→photo, Day→night, Thermal→color photos. Photo with missing pixels→inpainted photo. To more holistically evaluate the visual quality of the results, two methods are used. First, "real vs. fake" perceptual studies on Amazon Mechanical Turk(AMT) are run. For graphics problems like colourization and photo generation, plausibility to a human observer is often the ultimate goal. Therefore, tests on map generation, aerial photo generation, and image colourization using this approach are done.

Conclusion – The author method, with L1+cGAN loss, fooled participants on 22.5% of trials. This number can be further tuned with better state-of-the-art models such as Applying a conditional GAN to semantic segmentation. The conditional GAN scored similarly to the present L2 variants but fell short of other full methods, which fooled participants on 27.8% of trials in the respective experiments.

[3] Key concepts – Text to image synthesis, Generative adversarial networks

Research – The basic generative adversarial networks (GANs) tends to have the most variety in flower morphology, while other methods tend to generate more class-consistent images. In this work the authors developed a simple and effective model for generating images based on detailed visual descriptions. The authors aim to learn a mapping directly from words and characters to image pixels. The authors aim to further scale up the model to higher resolution images and add more types of text. The authors demonstrated that the model can synthesize many plausible visual interpretations of a given text caption. The manifold interpolation regularizer substantially improved the text to image synthesis on CUB. The authors showed disentangling of style and content, and bird pose and background transfer from query images onto text descriptions. Finally they demonstrated the generalizability of The approach to generating images with multiple objects and variable backgrounds with The results on MS-COCO datasets.

Methodology – The authors' approach is to train a deep convolutional generative adversarial network (DC-GAN) conditioned on text features encoded by a hybrid character-level convolutional recurrent neural network. Both the generator network G and the discriminator network D perform feed-forward inference conditioned on the text feature. They introduced a manifold interpolation regularizer for the GAN generator that significantly improves the quality of generated samples, including on held out zero shot categories on CUB

Conclusion – In future work, the authors aim to further scale up the model to higher resolution images and add more types of text. The authors aim to improve their model accuracy as well as run-time to achieve better results than other models such as deconvolutional neural networks, analogy pair encoding transformations etc.

[4] Key concepts – Generated adversarial nets, Deep Boltzmann machines, MNIST datasets

Research – In this work the authors introduce the conditional version of generative adversarial nets, which can be constructed by simply feeding the data, y, the authors wish to condition on to both the generator and discriminator. the authors show that this model can generate MNIST digits conditioned on class labels. the authors also illustrate how this model could be used to learn a multi-modal model, and provide preliminary examples of an application to image tagging

in which the authors demonstrate how this approach can generate descriptive tags which are not part of training labels.

Methodology – The authors trained a conditional adversarial net on MNIST images conditioned on their class labels, encoded as one-hot vectors. For the experiments the authors use MIR Flickr 25,000 dataset [10], and extract the image and tags features using the convolutional model and language model the authors described above. The discriminator is consisted of 500 and 1200 dimension ReLu hidden layers for word vectors and image features respectively and maxout layer with 1000 units and 3 pieces as the join layer which is fed to the one single sigmoid unit.

Conclusion - The results shown in this paper are extremely preliminary, but they demonstrate the potential of conditional adversarial nets and show promise for interesting and useful applications. Another obvious direction left for future work is to construct a joint training scheme to learn the language model.

[5] Key concepts – Large scale image recognition, local response normalization

Research – In this work the authors investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. The main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16-19 weight layers. These findings were the basis of their ImageNet Challenge 2014 submission, where their team secured the first and the second places in the localisation and classification tracks respectively. the authors also show that their representations generalise well to other datasets, where they achieve state-of-the-art results.

Methodology – Various state of the art models were evaluated, VGG (2 nets, multi-crop & dense eval.), VGG (1 net, multi-crop & dense eval.), VGG (ILSVRC submission, 7 nets, dense eval.), GoogLeNet (Szegedy et al, 2014) (1 net), GoogLeNet (Szegedy et al, 2014) (7 nets), MSRA (He et al, 2014) (11 nets), MSRA (He et al, 2014) (1 net), Clarifai (Russakovsky et al, 2014), Clarifai (Russakovsky et al, 2014) (1 net), Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets). The authors considered an ensemble of only two best-performing multi-scale models, which reduced the test error to 7.0% using dense evaluation and 6.8% using combined dense and multi-crop evaluation. The authors' best-performing single model achieves 7.1% error. In terms

of the single-net performance, the architecture achieves the best result (7.0% test error), outperforming a single GoogLeNet by 0.9%

Conclusion – The authors' results yet again confirm the importance of depth in visual representations. their result is also competitive with respect to the classification task winner (GoogLeNet with 6.7% error) and substantially outperforms the ILSVRC-2013 winning submission Clarifai, which achieved 11.2% with outside training data and 11.7% without it. The models can be further researched upon and improved to achieve better results than the current generation state-of-the -art models.

[6] Key concepts – Simultaneous detection and segmentation, Deep convolution networks

Research – Recognition algorithms based on convolutional networks (CNNs) typically use the output of the last layer as a feature representation. However, the information in this layer may be too coarse spatially to allow precise localization. On the contrary, earlier layers may be precise in localization but will not capture semantics. To get the best of both worlds, the authors define the hypercolumn at a pixel as the vector of activations of all CNN units above that pixel. Using hypercolumns as pixel descriptors, the authors show results on three fine-grained localization tasks: simultaneous detection and segmentation [22], where the authors improve state-of-the-art from 49.7 mean APr [22] to 60.0, keypoint localization, where the authors get a 3.3 point boost over [20], and part labeling, where the authors show a 6.6 point gain over a strong baseline. The authors have shown that the hypercolumn representation provides large gains in three different tasks.

Methodology – Gkioxari et al [20] fine-tuned a network for pose, person detection and action classification, and trained an SVM to assign a score to the keypoint predictions. A baseline system trained using the pipeline but with just the fc7 features performs significantly worse than the system, and is even worse than a HOG-based method

Constraints – The authors believe that hypercolumn representation might prove useful for other fine-grained tasks such as attribute or action classification. The model created by the authors does not specifically tune the attribute and action classification tasks. This research was left by the authors for future research.

[7] Key concepts – Convolutional neural network, Auto colourization, ReLU

Research – the authors present a novel technique to automatically colorize grayscale images that combines both global priors and local image features. the authors validate their approach with a user study and compare against the state of the art, where the authors show significant improvements. Furthermore, the authors demonstrate their method extensively on many different types of images, including black-and-white photography from over a hundred years ago, and show realistic colorizations.

Methodology— their approach uses a combination of global image priors, which are extracted from the entire image, and local image features, which are computed from small image patches, to colorize an image automatically. the authors have presented a novel architecture for the colorization of grayscale images by fusing both global and local information. While the baseline performs fairly poorly with roughly 70% of the images being considered natural, their approach has a median of 92.6%, which is close to the 97.7% of the ground truth. the authors train their model end-to-end on a large dataset for scene recognition with a joint colorization and classification loss that allows it to understand colors, and adapts the colors to the context of the image.

Conclusion – The main limitation of the method lies in the fact that it is data driven and thus will only be able to colorize images that share common properties with those in the training set. the authors have trained their model with a very large diverse set of both indoor and outdoor scene images in order to mitigate this. However, this does not contain, for example, human-created images. If the authors wish to evaluate on significantly different types of images, it would be necessary to train a new model for the new images. Transferring the style of semantically unrelated images, such as those of an aquarium to an image of a baseball stadium, do not generally give realistic results, although it is not clear what result to expect in this case. This limitation mainly arises from the fact that their approach only uses the grayscale of both images and does not modify the luminance of the output image, as style transfer is not the main focus of this work. Colorization is also an inherently ambiguous problem: is a man's shirt red or green? This has no unique solution. By learning from the data, their model will mainly use the dominant colors that it has learned, there is no explicit way for the user to control the colors besides manually setting different global features. It is likely that there is a way to handle this by adding an additional optimization on the colorization process itself. However, this possibility is not explored in this paper.

[8] Key concepts – Image segmentation, automatic colourization, contour detection

Research – the authors develop a fully automatic image colorization system. their approach leverages recent advances in deep networks, exploiting both low-level and semantic representations. As many scene elements naturally appear according to multimodal color distributions, the authors train their model to predict per-pixel color histograms. This intermediate output can be used to automatically generate a color image, or further manipulated prior to image formation. On both fully and partially automatic colorization tasks, the authors outperform existing methods. the authors also explore colorization as a vehicle for self-supervised visual representation learning.

Methodology – The authors frame the colorization problem as learning a function $f : X \rightarrow Y$. Prior to training for colorization, the authors further fine-tune the network for one epoch on the ImageNet classification task with grayscale input. By virtue of comparing to ground-truth color images, quantitative colorization metrics can penalize reasonable, but incorrect, color guesses for many objects more than jarring artifacts. This makes qualitative results for colorization as important as quantitative; the authors report both. Two novel contributions enable this progress: a deep neural architecture that is trained end-to-end to incorporate semantically meaningful features of varying complexity into colorization, and a color histogram prediction framework that handles uncertainty and ambiguities inherent in colorization while preventing jarring artifacts.

Conclusion – The author highlights how the colorization models are a promising avenue for self-supervised learning. The main constraints are associated with using the VGG-16 pretrained network adapted to the ImageNet. This makes the process very inflexible and introduces preset biases and variances for a variety of problems. The added histogram also further increases the total execution time and requirements for the final model

[9] Key concepts – Support vector regression, Principle component analysis, automatic colourization

Research – the authors aim to color greyscale images automatically, without any manual intervention. The color proposition could then be interactively corrected by user-provided color landmarks if necessary. The resulting algorithm is fast, designed to be more robust to texture

noise, and is above all able to deal with ambiguity, in contrary to previous approaches. the authors proposed a new approach for automatic image colorization, which does not require any intervention by the user, except from the choice of relatively similar color image. the authors stated the problem mathematically as an optimization problem with an explicit energy to minimize. Since it deals with multimodal probability distributions until the final step, this approach makes better use of the information that can be extracted from images by machine learning tools

Methodology – The contribution of their framework is that the authors deal directly with multimodality and estimate, for each pixel of the image to be colored, the probability distribution of all possible colors, instead of choosing the most probable color at the local level. the authors also predict the expected variation of color at each pixel, thus defining a non-uniform spatial coherency criterion. the authors then use graph cuts to maximize the probability of the whole colored image at the global level. the authors work in the L-a-b color space in order to approximate the human perception of distances between colors, and the authors use machine learning tools to extract as much information as possible from a dataset of colored examples.

Conclusion - The authors' contribution is both theoretical and practical. The authors proposed a new approach for automatic image colorization, which does not require any intervention by the user, except from the choice of relatively similar color images. The fact that the authors solve directly the problem at the global level, with the help of graph cuts, makes the framework more robust to noise and local prediction errors. It makes it possible to resolve large scale ambiguities which previous approaches could not. This multi-modality framework is not specific to image colorization and could be used in any prediction task on images.

[10] Key concepts – LEARCH, automatic image colourization, Gaussian random field

Research – We describe a method that learns to colorize grey-level images. We show how to incorporate a desired color histogram into the objective function, and that doing so can lead to further improvements in result. We show that possessing a scene label at run-time always provides a target histogram that results in improved quantitative performance performance; this scene label could come from an oracle, from application logic, or from a scene classifier applied to the grey-level image. We model the desired target histogram(t) as Gaussian mixture model (GMM) obtained using the EM algorithm. We propose a method to predict colorization using an

objective automatically learned by LEARCH. Our method significantly outperforms the best baseline we are aware of, in the first quantitative colorization experiments we are aware of. We demonstrate that the method produces spatially coherent colorization, and when augmented with histogram correction produces visually appealing and convincing colorizations

Methodology – Their method exploits a LEARCH framework to train a quadratic objective function in the chromaticity maps, comparable to a Gaussian random field. The coefficients of the objective function are conditioned on image features, using a random forest. The objective function admits correlations on long spatial scales, and can control spatial error in the colorization of the image. Images are then colorized by minimizing this objective function. They perform colorization on 6 scene categories of the SUN dataset, viz. beach, castle, outdoor, kitchen, living room, bedroom. The authors chose 3 indoor and 3 outdoor categories with maximum number of images. All images are rescaled to have height of 256, with aspect ratio maintained. 2-Channel Error. For each scene category, they randomly select 40 color/greylevel image pairs as training data, 20 image pairs for validation and 40 grey-level images for testing. For scene independent training, they merge the training images of all the 6 categories together. They perform a parameter search on validation and use the optimal parameters in test (Section 4.2). The remaining images in each category are used as a database to obtain the top-k matching images for generating average color image. There are two ways to perform automatic colorization, either they use scene independent LEARCH or they generate scene labels using scene classification and pick the appropriate scene specific regressor. Scene classification LEARCH with histogram correction outperforms scene independent LEARCH

Conclusion - The method produces good color images as output, in fact use of ground-truth histogram allows them to output strikingly similar looking images to the ground truth. The output color images with LEARCH followed by correction with mean histogram also show good resemblance to ground-truth. They are free from spatial oddities, unlike Welsh et al and average color image. Generally large regions are assigned close to ground-truth colors, but smaller regions/objects are assigned spatially coherent but incorrect colors. This is likely because they are not sampled frequently. They demonstrate that the method produces spatially coherent colorization, and when augmented with histogram correction produces visually appealing and convincing colorizations. The method performs best when scene information is available from an

oracle, but our fully automated approach, which uses scene classification, produces near optimal results

[11] Key concepts – Chromatic information

Research – We introduce a general technique for "colorizing" greyscale images by transferring color between a source, color image and a destination, greyscale image. Although the general problem of adding chromatic values to a greyscale image has no exact, objective solution, the current approach attempts to provide a method to help minimize the amount of human labor required for this task. Rather than choosing RGB colors from a palette to color individual components, we transfer the entire color "mood" of the source to the target image by matching luminance and texture information between the images. We choose to transfer only chromatic information and retain the original luminance values of the target image. Further, the procedure is enhanced by allowing the user to match areas of the two images with rectangular swatches. We show that this simple technique can be successfully applied to a variety of images and video, provided that texture and luminance are sufficiently distinct. The images generated demonstrate the potential and utility of our technique in a diverse set of application domains

Methodology – While standard methods accomplish this task by assigning pixel colors via a global color palette, our technique empowers the user to first select a suitable color image and then transfer the color mood of this image to the greylevel image at hand. We have intentionally kept the basic technique simple and general by not requiring registration between the images or incorporating spatial information. Our technique can be made applicable to a larger class of images by adding a small amount of user guidance. In this mode, the user first transfers the desired color moods from a set of specified swatch regions in the color image to a set of corresponding swatch regions in the greyscale image. Then, in the second and final stage of the colorization process, the colorized swatches are employed, using a texture synthesis-like method, to colorize the remaining pixels in the greyscale image. Currently, the L2 distance is used to measure texture similarity within the image. In the future we believe the technique can be substantially improved by using a more sophisticated measure of texture similarity

Conclusion - Our technique of employing an example color image to colorize a greylevel image is particularly attractive in light of the growing sophistication of internet image search engines and the emergence of centralized and indexable image collections which can be used to easily

locate suitable color images. Finally, one could also utilize a database of basis texture swatches for the initial color transfer in the user-guided stage of the colorization process.

[12] Key concepts – open boundary, gabor wavelet filter

Research – This paper proposes a novel colorization technique that propagates color over regions exhibiting pattern-continuity as well as intensity-continuity. The proposed method works effectively on colorizing black-and-white manga which contains intensive amount of strokes, hatching, halftoning and screening. Such fine details and discontinuities in intensity introduce many difficulties to intensity-based colorization methods. Once the user scribbles on the drawing, a local, statistical based pattern feature obtained with Gabor wavelet filters is applied to measure the pattern-continuity. The boundary is then propagated by the level set method that monitors the pattern-continuity. Regions with open boundaries or multiple disjointed regions with similar patterns can be sensibly segmented by a single scribble. With the segmented regions, various colorization techniques can be applied to replace colors, colorize with stroke preservation, or even convert pattern to shading. Several results are shown to demonstrate the effectiveness and convenience of the proposed method.

Methodology –

This paper proposes a novel colorization technique that propagates color over regions exhibiting pattern-continuity as well as intensity-continuity. The proposed method works effectively on colorizing black-and-white manga which contains intensive amount of strokes, hatching, halftoning and screening. Such fine details and discontinuities in intensity introduce many difficulties to intensity-based colorization methods. Once the user scribbles on the drawing, a local, statistical based pattern feature obtained with Gabor wavelet filters is applied to measure the pattern-continuity. The boundary is propagated by the level set method that monitors the pattern-continuity. Regions with open boundaries or multiple disjointed regions with similar patterns can be sensibly segmented by a single scribble. Various colorization techniques can be applied to replace colors, colorize with stroke preservation, or even convert pattern to shading.

Several results are shown to demonstrate the effectiveness and convenience of the proposed method. The pleasantville post production team that focussed on the absence of color.

Conclusion - With the segmented regions, various colorization techniques can be applied to replace colors, colorize with stroke preservation, or even convert pattern to shading. Several results are shown to demonstrate the effectiveness and convenience of the proposed method.

[13] Key concepts – grayscale image, object boundary, colorization algorithm

Research – Colorization is a computer-assisted process for adding colors to grayscale images or movies. It can be viewed as a process for assigning a three-dimensional color vector (YUV or RGB) to each pixel of a grayscale image. In previous works, with some color hints the resultant chrominance value varies linearly with that of the luminance. However, it is easy to find that existing methods may introduce obvious color bleeding, especially, around region boundaries. It then needs extra human-assistance to fix these artifacts, which limits its practicability. Facing such a challenging issue, we introduce a general and fast colorization methodology with the aid of an adaptive edge detection scheme. By extracting reliable edge information, the proposed approach may prevent the colorization process from bleeding over object boundaries. Next, integration of the proposed fast colorization scheme to a scribble-based colorization system, a modified color transferring system and a novel chrominance coding approach are investigated. In our experiments, each system exhibits obvious improvement as compared to those corresponding previous works.

Methodology – It can be viewed as a process for assigning a three-dimensional color vector (YUV or RGB) to each pixel of a grayscale image. In previous works, with some color hints the resultant chrominance value varies linearly with that of the luminance. However, it is easy to find that existing methods may introduce obvious color bleeding, especially, around region boundaries. It then needs extra human-assistance to fix these artifacts, which limits its practicability. Facing such a challenging issue, we introduce a general and fast colorization methodology with the aid of an adaptive edge detection scheme

Conclusion - By extracting reliable edge information, the proposed approach may prevent the colorization process from bleeding over object boundaries. Next, integration of the proposed fast colorization scheme to a scribble-based colorization system, a modified color transferring system and a novel chrominance coding approach are investigated. In our experiments, each system exhibits obvious improvement as compared to those corresponding previous works.

[14] Key concepts – user intervention, optimization problem, segmentation

Research – Colorization of classic motion pictures has generated much controversy [Cooper 1991], which partially accounts for the fact that not many of these movies have been colorized to date. However, there are still massive amounts of black and white television shows that could be colorized: the artistic controversy is often irrelevant here, while the financial incentives are substantial, as was succinctly pointed out by Earl Glick1 in 1984: "You couldn't make Wyatt Earp today for $1 million an episode. But for $50,000 a segment, you can turn it into color and have a brand new series with no residuals to pay" [Burns]. Colorization of still images also appears to be a topic of considerable interest among users of image editing software, as evidenced by multiple colorization tutorials on the World Wide Web.

Methodology – In this paper we present a simple colorization method that requires neither precise image segmentation, nor accurate region tracking. Our method is based on a simple premise: neighboring pixels in space-time that have similar intensities should have similar colors. We formalize this premise using a quadratic cost function and obtain an optimization problem that can be solved efficiently using standard techniques. In our approach an artist only needs to annotate the image with a few color scribbles, and the indicated colors are automatically propagated in both space and time to produce a fully colorized image or sequence. We demonstrate that high quality colorizations of stills and movie clips may be obtained from a relatively modest amount of user input. CR Categories: I.4.9 [Image Processing and Computer Vision]: Applications; Keywords: colorization, recoloring, segmentation

Conclusion - The results shown here were all obtained using the correlation based window (equation 3, or equivalently using the local linearity assumption). The mean and variance μ, σ for each pixel were calculated by giving more weight to pixels with similar intensities. Visually similar results were also obtained with the Gaussian window (equation 2). For still images we used Matlab's built in least squares solver for sparse linear systems, and for the movie sequences we used a multigrid solver [Press et al 1992]. Using the multigrid solver, the run time was approximately 15 seconds per frame. The threshold T in equation 4 was set to 1 so that the window used was $3 \times 3 \times 3$.

[15] Key concepts – colorization algorithm

Research – In this paper, drawing intuition from the Turing test, we propose using adversarial training for open-domain dialogue generation: the system is trained to produce sequences that are indistinguishable from human-generated dialogue utterances. We cast the task as a reinforcement learning (RL) problem where we jointly train two systems, a generative model to produce response sequences, and a discriminator---analagous to the human evaluator in the Turing test--- to distinguish between the human-generated dialogues and the machine-generated ones. The outputs from the discriminator are then used as rewards for the generative model, pushing the system to generate dialogues that mostly resemble human dialogues.

Methodology – In addition to adversarial training we describe a model for adversarial {\em evaluation} that uses success in fooling an adversary as a dialogue evaluation metric, while avoiding a number of potential pitfalls. Experimental results on several metrics, including adversarial evaluation, demonstrate that the adversarially-trained system generates higher-quality responses than previous baselines.

Conclusion - The outputs from the discriminator are then used as rewards for the generative model, pushing the system to generate dialogues that mostly resemble human dialogues.

[16] Key concepts – Deep Learning, grayscale image

Research – The promise of deep learning is to discover rich, hierarchical models [2] that represent probability distributions over the kinds of data encountered in artificial intelligence applications, such as natural images, audio waveforms containing speech, and symbols in natural language corpora. So far, the most striking successes in deep learning have involved discriminative models, usually those that map a high-dimensional, rich sensory input to a class label [14, 20]. These striking successes have primarily been based on the backpropagation and dropout algorithms, using piecewise linear units [17, 8, 9] which have a particularly well-behaved gradient . Deep generative models have had less of an impact, due to the difficulty of approximating many intractable probabilistic computations that arise in maximum likelihood estimation and related strategies, and due to difficulty of leveraging the benefits of piecewise linear units in the generative context. We propose a new generative model estimation procedure that sidesteps these difficulties.

Methodology – In the proposed adversarial nets framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.

Conclusion - A conditional generative model p(x | c) can be obtained by adding c as input to both G and D. 2. Learned approximate inference can be performed by training an auxiliary network to predict z given x. This is similar to the inference net trained by the wake-sleep algorithm [15] but with the advantage that the inference net may be trained for a fixed generator net after the generator net has finished training. 3. One can approximately model all conditionals p(xS | x6S) where S is a subset of the indices of x by training a family of conditional models that share parameters. Essentially, one can use adversarial nets to implement a stochastic extension of the deterministic MP-DBM [10]. 4. Semi-supervised learning: features from the discriminator or inference net could improve performance of classifiers when limited labeled data is available. 5. Efficiency improvements: training could be accelerated greatly by devising better methods for coordinating G and D or determining better distributions to sample z from during training.

[17] Key concepts – Support Vector Regression, Color Variation, Local Description

Research – We aim to color greyscale images automatically, without any manual intervention. The color proposition could then be interactively corrected by user-provided color landmarks if necessary. Automatic colorization is nontrivial since there is usually no one-to-one correspondence between color and local texture. The contribution of our framework is that we deal directly with multimodality and estimate, for each pixel of the image to be colored, the probability distribution of all possible colors, instead of choosing the most probable color at the local level.

Methodology – We also predict the expected variation of color at each pixel, thus defining a non-uniform spatial coherency criterion. We then use graph cuts to maximize the probability of the whole colored image at the global level. We work in the L-a-b color space in order to

approximate the human perception of distances between colors, and we use machine learning tools to extract as much information as possible from a dataset of colored examples. The resulting algorithm is fast, designed to be more robust to texture noise, and is above all able to deal with ambiguity, in contrary to previous approaches.

Conclusion - The resulting algorithm is fast, designed to be more robust to texture noise, and is above all able to deal with ambiguity, in contrary to previous approaches**.**

[18] Key concepts – grayscale, deep learning

Research – This paper investigates into the colorization problem which converts a grayscale image to a colorful version. This is a very difficult problem and normally requires manual adjustment to achieve artifact-free quality. For instance, it normally requires human-labelled color scribbles on the grayscale target image or a careful selection of colorful reference images (e.g., capturing the same scene in the grayscale target image). Unlike the previous methods, this paper aims at a high-quality fully-automatic colorization method. With the assumption of a perfect patch matching technique, the use of an extremely large-scale reference database (that contains sufficient color images) is the most reliable solution to the colorization problem.

Methodology – The patch matching noise will increase with respect to the size of the reference database in practice. Inspired by the recent success in deep learning techniques which provide amazing modeling of large-scale data, this paper re-formulates the colorization problem so that deep learning techniques can be directly employed. To ensure artifact-free quality, a joint bilateral filtering based post-processing step is proposed. We further develop an adaptive image clustering technique to incorporate the global image information. Numerous experiments demonstrate that our method outperforms the state-of-art algorithms both in terms of quality and speed.

Conclusion - We further develop an adaptive image clustering technique to incorporate the global image information. Numerous experiments demonstrate that our method outperforms the state-of-art algorithms both in terms of quality and speed.

[19] Key concepts – grayscale images, deep patch-wise colorization

Research – To handle the colorization problem, we propose a deep patch-wise colorization model for grayscale images. Distinguished with some constructive color mapping models with complicated mathematical priors, we alternately apply two loss metric functions in the deep model to suppress the training errors under the convolutional neural network.

Methodology – To address the potential boundary artifacts, a refinement scheme is presented inspired by guided filtering. In the experiment section, we summarize our network parameters setting in practice, including the patch size, amount of layers and the convolution kernels. Our experiments demonstrate this model can output more satisfactory visual colorizations compared with the state-of-the-art methods. Moreover, we prove our method has extensive application domains and can be applied to stylistic colorization.

Conclusion - Our experiments demonstrate this model can output more satisfactory visual colorizations compared with the state-of-the-art methods. Moreover, we prove our method has extensive application domains and can be applied to stylistic colorization.

[20] Key concepts – self-supervised feature learning, gray-scale

Research – Given a grayscale photograph as input, this paper attacks the problem of hallucinating a plausible color version of the photograph. This problem is clearly underconstrained, so previous approaches have either relied on significant user interaction or resulted in desaturated colorizations. We propose a fully automatic approach that produces vibrant and realistic colorizations. We embrace the underlying uncertainty of the problem by posing it as a classification task and use class-rebalancing at training time to increase the diversity of colors in the result.

Methodology – The system is implemented as a feed-forward pass in a CNN at test time and is trained on over a million color images. We evaluate our algorithm using a "colorization Turing test," asking human participants to choose between a generated and ground truth color image. Our method successfully fools humans on 32% of the trials, significantly higher than previous methods. Moreover, we show that colorization can be a powerful pretext task for self-supervised

feature learning, acting as a cross-channel encoder. This approach results in state-of-the-art performance on several feature learning benchmarks.

Conclusion - Our method successfully fools humans on 32% of the trials, significantly higher than previous methods. Moreover, we show that colorization can be a powerful pretext task for self-supervised feature learning, acting as a cross-channel encoder. This approach results in state-of-the-art performance on several feature learning benchmarks.

[21] Key concepts – Image colorization

Research – Image colorization assigns a color to each pixel of a target grayscale image. Colorization methods can be roughly divided into two categories: scribble-based colorization and example-based colorization. The scribble-based methods typically require substantial efforts from the user to provide considerable scribbles on the target grayscale images. Manual segmentation cues are hard to obtain as the target grayscale image may contain multiple complex objects. These methods share the same limitation − their performance highly depends on the selected reference image(s). This paper aims at a high-quality fully-automatic colorization method.

Methodology – The state-of-the-art methods are very slow because they have to find the most similar pixels from massive candidates. The training of neural networks is slow especially when the database is large, colorizing a 256×256 grayscale image using the trained neural network assemble takes only 6.780 seconds in Matlab. The proposed colorization neural network assemble is trained on 2344 images from the SIFT Flow database. The neural network has an input layer, three hidden layers and one output layer. According to the experiments, using more hidden layers cannot further improve the colorization results. There are a total of 114 neurons in the input layer. This paper perceptually sets the number of neurons in the hidden layer to half of that in the input layer and 2 neurons in the output layer. The authors use gist feature [29] as the global image descriptor in the experiment

Conclusion - This paper presents a novel, fully-automatic colorization method using deep neural networks to minimize user effort and the dependence on the example color images. Informative yet discriminative features including patch feature, DAISY (b) The authors' method (c) Ground truth feature and a new semantic feature are extracted and serve as the input to the neural network. The output chrominance values are further refined using joint bilateral filtering to ensure artifact-

free colorization quality. Numerous experiments demonstrate that the method outperforms the state-of-art algorithms both in terms of quality and speed.

[22] Key concepts – Sketch colorization

Research – In the field of animation drawing, it usually creates grayscale line sketch at first and coloring, processing and adjustment. Deep learning has flourished, some method has been proposed which automatic coloring or according to the user's reference hint coloring by using neural networks. After generative adversarial networks (GANs) [4] proposed, several sketch automatic coloring systems by using GANs, like Paintschainer [8], Style2paints [7] have achieved some good results.

Methodology – SYSTEM STRUCTURE Given anime line sketch and corresponding color hints, the authors train a deep residual neural network can let anime line sketch be colored by according color hints. In order to allow the network learning to color according to the color hint given by the user, the authors created a discriminator D to learn identify generated image is good or bad, and the authors added color hint input in 3st level of the generator G. Input anime line sketches and the random color hints of the real color images to the generator G, to generate a coloring image. The real color image and coloring image output by colorization generator will input to the discriminator D to calculate loss, when backward for discriminator D, the label of sigmoid cross entropy of the real color image will be 1 , of the coloring image will be 0, on the other hand when backward for generator G the label of sigmoid cross entropy of coloring image will be 1, expect that the network learning is colored according to the given color hint.

Conclusion - The generator G input is D={S, T, H}, where S is a 256*256 anime line sketch, T is the 256*256 the real color anime image corresponding to S, H is the 64*64 color hints that gerented by 4 times downsampled T and randomly selects the color points in T. The generator G input is D= {S, H}, S is the anime line sketch of 256*256 size, H is the color hint map given by the user, the size is 256*256, generate color image. C. Experimental results This experiment uses 4 line sketches as the test data, and user draws the color hint corresponding to the line sketch. This study generated a color image that was consistency or similar to the color hint given by the user, the quality of color required to be improved in the future as well as the better consistency with the color hint given by the user.

[23] Key concepts – Automatic colorization, gray scale

Research – We aim to color greyscale images automatically, without any manual intervention. The color proposition could then be interactively corrected by user-provided color landmarks if necessary. Automatic colorization is nontrivial since there is usually no one-to-one correspondence between color and local texture. The contribution of our framework is that we deal directly with multimodality and estimate, for each pixel of the image to be colored, the probability distribution of all possible colors, instead of choosing the most probable color at the local level. We also predict the expected variation of color at each pixel, thus defining a nonuniform spatial coherency criterion. We then use graph cuts to maximize the probability of the whole colored image at the global level. We work in the L-a-b color space in order to approximate the human perception of distances between colors, and we use machine learning tools to extract as much information as possible from a dataset of colored examples. The resulting algorithm is fast, designed to be more robust to texture noise, and is above all able to deal with ambiguity, in contrary to previous approaches.

Methodology – We first model the basic quantities to be considered in the image colorization problem. Let I denote a greyscale image to be colored, p the location of one particular pixel, and C a proposition of colorization, that is to say an image of same size as I but whose pixel values C(p) are in the standard RGB color space. Since the greyscale information is already given by I(p), we should add restrictions on C(p) so that computing the greyscale intensity of C(p) should give I(p) back. Thus the dimension of the color space to be explored is intrinsically 2 rather than 3. We present in this section the model chosen for the color space, our way to discretize it for further purposes, and how to express continuous probability distributions of colors out of such a discretization. We also present the feature space used for the description of greyscale patches

Conclusion - Our colorization approach outperforms [5] and [6], whose examples contain only few colors and lack spatial coherency. Our process requires less or similar intervention than [7] but can handle more ambiguous cases and more texture noise. The computational time needed is also very low, enabling real-time user interaction since a re-colorization with new color landmarks lasts a fraction of second. The automatic colorization results should not be compared to those obtained by user-helped approaches since we do not benefit from such decisive

information. Nevertheless, even with such an handicap, our results clearly compete with the state of the art of colorization with few user scribbles

[24] Key concepts – large technical professional organization

Research – The sketch plays an important role in the animation industry. Auto or semi-auto colorizing sketches will improve animators' efficiency and reduce the production costs. In this paper, we propose a semi-auto sketch colorization method based on conditional generative adversarial networks, which can support user interaction by adding scribbles to guide the colorization process.

Methodology – In addition, we apply a pre-model to extract high-level features of sketches in order to make good use of sketches' unique texture information. Furthermore, the loss function in our method is specially designed that can reduce blend and overflow in the result. At last, we use joint bilateral filter to smooth the output and generate a cleaner and vivid coloring sketch. Experimental results show that every module in our method can make a contribution to the final results. Moreover, the comparison with PaintsChainer demonstrates that our method can avoid large areas of leakage in the background and have cleaner skin for characters.

Conclusion - Experimental results show that every module in our method can make a contribution to the final results. Moreover, the comparison with PaintsChainer demonstrates that our method can avoid large areas of leakage in the background and have cleaner skin for characters.

[25] Key concepts – first-order gradient-based optimization

Research – We introduce Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients.

Methodology – The hyper-parameters have intuitive interpretations and typically require little tuning. Some connections to related algorithms, on which Adam was inspired, are discussed. We also analyze the theoretical convergence properties of the algorithm and provide a regret bound on the convergence rate that is comparable to the best known results under the online convex optimization framework. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. Finally, we discuss AdaMax, a variant of Adam based on the infinity norm.

Conclusion - Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. Finally, we discuss AdaMax, a variant of Adam based on the infinity norm.

[26] Key Concept – Using dropout to improve performance of neural networks on supervised learning

Research – Backpropagation Dropout neural networks can be trained using stochastic gradient descent in a manner similar to standard neural nets. The only difference is that for each training case in a mini-batch, we sample a thinned network by dropping out units. Forward and backpropagation for that training case are done only on this thinned network. The gradients for each parameter are averaged over the training cases in each mini-batch. Any training case which does not use a parameter contributes a gradient of zero for that parameter. Many methods have been used to improve stochastic gradient descent such as momentum, annealed learning rates and L2 weight decay. Those were found to be useful for dropout neural networks as well. trained dropout neural networks for classification problems on data sets in different domains. We found that dropout improved generalization performance on all data sets compared to neural networks that did not use dropout.

Methodology – Standard Neural Net (Simard et al, 2003) SVM Gaussian kernel Dropout NN Dropout NN Dropout NN + max-norm constraint Dropout NN + max-norm constraint Dropout NN + max-norm constraint Dropout NN + max-norm constraint Dropout NN + max-norm constraint (Goodfellow et al, 2013), 3 layers, 1024 units 3 layers, 1024 units 3 layers, 1024 units 3 layers, 2048 units 2 layers, 4096 units 2 layers, 8192 units 2 layers, (5 × 240). Dropout nets pretrained with stacks of RBMs and Deep Boltzmann Machines also give improvements, DBM—pretrained dropout nets achieve a test error of 0.79% which is the best performance ever reported

for the permutation invariant setting. We note that it possible to obtain better results by using 2-D spatial information and augmenting the training set with distorted versions of images from the standard training set. We demonstrate the effectiveness of dropout in that setting on more interesting data sets.

Conclusion - Dropout is a technique for improving neural networks by reducing overfitting. Standard backpropagation learning builds up brittle co-adaptations that work for the training data but do not generalize to unseen data. Random dropout breaks up these co-adaptations by making the presence of any particular hidden unit unreliable. This technique was found to improve the performance of neural nets in a wide variety of application domains including object classification, digit recognition, speech recognition, document classification and analysis of computational biology data. This suggests that dropout is a general technique and is not specific to any domain. Methods that use dropout achieve state-of-the-art results on SVHN, ImageNet, CIFAR-100 and MNIST. Dropout considerably improved the performance of standard neural nets on other data sets as well.

[27] Key Concept – Region-based Convolutional Neural Networks

Research – The Region-based Convolutional Network method (RCNN) [9] achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN, however, has notable drawbacks: 1. Training is a multi-stage pipeline. R-CNN first finetunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned. Training is expensive in space and time. For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk. With very deep networks, such as VGG16, this process takes 2.5 GPU-days for the 5k images of the VOC07 trainval set. These features require hundreds of gigabytes of storage. Object detection is slow. At test-time, features are extracted from each object proposal in each test image. Detection with VGG16 takes 47s / image (on a GPU).

Methodology – R-CNN is slow because it performs a ConvNet forward pass for each object proposal, without sharing computation. Spatial pyramid pooling networks (SPPnets) [11] were proposed to speed up R-CNN by sharing computation. The SPPnet method computes a

convolutional feature map for the entire input image and then classifies each object proposal using a feature vector extracted from the shared feature map. Features are extracted for a proposal by maxpooling the portion of the feature map inside the proposal into a fixed-size output (e.g., 6 × 6). Multiple output sizes are pooled and then concatenated as in spatial pyramid pooling [15]. SPPnet accelerates R-CNN by 10 to 100× at test time. Training time is also reduced by 3× due to faster proposal feature extraction. First, the last max pooling layer is replaced by a RoI pooling layer that is configured by setting H and W to be compatible with the net's first fully connected layer (e.g., H = W = 7 for VGG16). Second, the network's last fully connected layer and softmax (which were trained for 1000-way ImageNet classification) are replaced with the two sibling layers described earlier (a fully connected layer and softmax over K + 1 categories and category-specific bounding-box regressors). Third, the network is modified to take two data inputs: a list of images and a list of RoIs in those images.

Conclusion - This paper proposes Fast R-CNN, a clean and fast update to R-CNN and SPPnet. In addition to reporting state-of-theart detection results, we present detailed experiments that we hope provide new insights. Of particular note, sparse object proposals appear to improve detector quality. This issue was too costly (in time) to probe in the past, but becomes practical with Fast R-CNN. Of course, there may exist yet undiscovered techniques that allow dense boxes to perform as well as sparse proposals. Such methods, if developed, may help further accelerate object detection.

[28] Key Concept – Shot detection and frame-by-frame colorization

Research – Since the human visual system is sensitive to colors rather than gray shades, the authors aim to emphasize the appearance of black and white movies and recolor them to obtain near natural colored movies that look like their original colors. The goal of the research is to implement a powerful automatic coloring system that is suitable for coloring movies with high quality colors and in fast time as possible .The authors' proposed system based on colorizing the movie shot by shot rather than frame by frame, so that different techniques are presented like shot cut detection , motion estimation, similarity features between images and colorization.

Methodology – The system results assured the vision of the possibility of coloring old movies in short time and high quality coloring. The colorization process for a shot of 44 frames in the classic movie "Esmaiel Yaseen Tarazan" took about 2 minutes and 32 seconds to colorize the shot using

motion vector. The system results assured the vision of the possibility of coloring old movies in short time and high quality coloring. The colorization process for a shot of 44 frames in the classic movie "Esmaiel Yaseen Tarazan" took about 2 minutes and 32 seconds to colorize the shot using motion vector. The coloring results appear to be satisfactory with respect to the first frame, key frame.

Conclusion - The proposed system presents various technologies dealing with gray and color images processing as well as video processing. The outcome of the search provides a tangible significant contribution to the culture, the IT industry or to another industry (Film industry) that is IT enabled since there is wealth of old historic pictures and movies. In order to increase the demand on these pictures or movies, a need to color them is growing. The outcome of the research is a prototype that utilizes the IT technologies to provide specific benefits to solution stakeholders (Film industry) vs. traditional slow, costly and manual methods.

[29] Key Concept – Automatic colorization using Generative adversarial networks

Research – In this paper, we focus on automatically colorizing single grayscale image without manual interventions. Most of existing methods tried to accurately restore unknown ground-truth colors and require paired training data for model optimization. However, the ideal restoration objective and strict training constraints limited their performance. Inspired by CycleGAN, we formulate the process of colorization as image-to-image translation and propose an effective colorCycleGAN solution. High-level semantic identity loss and low-level color loss are additionally suggested for model optimization. Our method allows using unpaired images for training and direct prediction in rgb color space, which makes training data collection much easier and more general. We train our model on randomly selected PASCAL VOC 2007 images. All ablation study on loss function and comparisons with state-of-the-art methods are performed on grayscale SUN data. The experiment results show that our improvements on training loss could achieve better content consistence and generate better reasonable colors with less artifacts. Moreover, due to the bidirectional nature of our model, our proposed method provides a by-product that gives an excellent alternative way on color image graying.

Methodology – The authors randomly select 2000 images from the Pascal VOC 2007 dataset [19] for training, of which 1000 images as colorful images, and 1000 images for grayscale processing.

Color images and grayscale images are not required to have same contents. All the training images are resized and cropped to 256×256. The authors test the trained model on SUN dataset

Conclusion - The authors have proposed an effective colorization method for grayscale images. The authors' approach could be trained without paired image data and color space transformation. The authors' introduced semantic identity loss and color loss enable generation network to get more reasonable colorful images. The experimental results show that the proposed method can achieve better colorization performance than several popular stat-of-the-art methods. The authors want to study a video auto-colorization model. The content continuity and color consistency between successive frames are the considerations in the future.

[30] Key Concept – Stochastic gradient descent

Research – Stochastic and online gradient descent methods have proved to be extremely useful for solving largescale machine learning problems [1, 2, 3, 4]. The key idea behind these algorithms is to prove a general regret bound in terms of an arbitrary sequence of non-increasing learning rates and the full sequence of gradients, and to define an adaptive method for choosing the learning rates as a function of the gradients seen so far, so as to minimize the final bound when the learning rates are plugged in. The authors extend this idea to the parallel setting, by developing a general regret bound that depends on both the gradients and the exact update delays that occur. In addition to providing an adaptive regret bound, the authors demonstrate excellent empirical performance. The authors' goal is to find an efficiently implementable algorithm that achieves comparable results in practice and matches this regret bound.

Methodology – The authors study the performance of both hypothetical algorithms and AdaptiveRevision on two realworld medium-sized datasets. The authors found this had a negligible impact; in the plots above, AdaptiveRevision$*$ denotes the algorithm without this check. With this improvement AdaptiveRevision stores three numbers per coefficient, versus the two stored by AsyncAdagrad DA or GD. The first two patterns satisfy InOrder, but the third does not.

Conclusion - Experimental results show when the delays grow large (1000 updates or more), the new algorithms perform significantly better than standard adaptive gradient methods. The authors have demonstrated that adaptive tuning and revision of per-coordinate learning rates for distributed gradient descent can significantly improve accuracy as the update delays become large. The

analysis method is novel, but is somewhat indirect; an interesting open question is finding a general analysis framework for algorithms of this style. Such an analysis would remove the technical need for the InOrder assumption, and allow for the analysis of AdaptiveRevision variants of OGD with Projection and Dual Averaging

[31] Key Concept – Digital Image Forensics

Research – This chapter discusses counter-forensics, the art and science of impeding or misleading forensic analyses of digital images. Forensic analyses are possible if the functions acquire and process leave identifying traces in the resulting images. A digital image forensics algorithm is given by a function decide : $I \rightarrow C$ that assigns an image $I \in I$ to a class $C \in C$. For a given image $I = I(k)$, counter-forensics aims at preventing the assignment to the image's class Ck. By suppressing or counterfeiting identifying traces, the counterfeiter creates a counterfeit $J = J(l)$ with the intention to let it appear like an authentic member of an alternative class $Cl \in C$, $l = k$, when presented to the forensic investigator's function decide.

Methodology – A digital image forensics algorithm decide is vulnerable to a counterforensic attack if for a given image $I = \text{generate}(\theta)$. Counterfeiting the source of an authentic image the author involves a single application of a counter-forensic attack $I \rightarrow I$, possibly with the additional requirement that a specific target class C. A counter-forensic attack is ε-reliable against all digital image forensics algorithms on a class space C if for each pair $(C0, C1) \in C \times C$, $I(Ck) \sim PCk$, $DKL\ PC0\ , PC1 \leq \varepsilon$. The robustness of a digital image forensics algorithms is defined by its reliability under legitimate post-processing. As a counter-forensic technique, legitimate post-processing does not require much knowledge of the image generation process. The security of a digital image forensics algorithm is defined by its reliability to detect intentionally concealed illegitimate post-processing. Counterfeiters attacking security properties exploit specific knowledge about and shortcomings of the image model used by forensic investigators. Post-processing attacks modify an image I(k) such that the resulting counterfeit I(l) does not reveal traces of the original class Ck anymore.

Conclusion - Post-processing attacks suppress and/or synthesize identifying traces subsequent to the original image generation process. Integrated attacks directly replace the original process with a counter-forensic variant Because integrated methods obviously require deep knowledge of the

image generation process, they do not address robustness issues of forensic algorithms by definition. A mere combination of all known targeted counter-forensic techniques does not yield a universal attack; at least when typical low-dimensional image models are employed, which ignore the interference and interdependence of different attacks. Certain forensic techniques test the authenticity of images by verifying the integrity of traces of the common image acquisition or processing chain.

[32] Key Concept – Adversarial Classifier Reverse Engineering

Research – Security-oriented applications of signal processing are receiving increasing attention. It is the aim of this paper to review the most recent advances in the fields where signal processing designers have to cope with the presence of an adversary, highlighting the similarities between the existing approaches, and provide a unitary view for one of the most commonly encountered problems, namely binary decision. After the general view the authors gave in the previous section, the authors delve into one specific problem, namely adversarial binary decision, showing how the advances made in various fields fit into a unique framework A similar approach for the case of linear boundaries and l1-norm distances has been proposed in the context of machine learning for spam filtering in the so-called Adversarial Classifier Reverse Engineering (ACRE) method.

Methodology – Assume a gradient descent-based algorithm to solve the closest-point problem is used; the designer will be interested in creating a decision boundary that leads to local minima, where the search can get stuck. If the authors assume that the training sequence used by the defender is not known to the attacker, it introduces into the picture the possibility of making the decision regions partially depend on a secret, getting closer to the typical scenario in watermarking. One of the drawbacks of assuming rational adversaries is that the solutions generally lead to very conservative designs of the decision function which in turn yield a bad performance even for non-malicious users

Conclusion - To conclude, we expect that in the near future a general framework for Adv-SP will be developed, by combining elements of game, detection, machine learning, optimization and complexity theories. We envisage that Adv-SP will become a stimulating and challenging field whose developments will immediately find a vast number of applications.

[33] Key Concept – Forensic Image Manipulation

Research – Legal, scientific, and news media organizations rely on digital images to make critical decisions or to use as photographic evidence of specific events. An image forger can alter a digital image in a visually realistic manner. To avoid both embarrassment and legal ramifications, many of these organizations desire some means of identifying image alterations and verifying image authenticity. The field of digital image forensics has been born

Methodology – The authors' simulation results show that aside from exceptional cases, all of the detection methods are able to correctly detect the use of their designated image processing operation with a probability of 99% given a false alarm probability of 7% or less. Results comparable to the previous experiment were achieved for images previously compressed using quality factors of 50 or greater. For these quality factors, a of 99% was achieved at a of 3.7% or less. The authors' simulation results show that aside from exceptional cases, each of the proposed techniques achieved a of 99% with a of 7% or less.

Conclusion - The authors proposed a set of digital image forensic techniques capable of detecting global and local contrast enhancement, identifying the use of histogram equalization, and detection of the global addition of noise to a previously JPEG-compressed image In each of these techniques, detection depends upon the presence or absence of an intrinsic fingerprint introduced into an image's histogram by a pixel value mapping. The authors proposed a technique which detects the global addition of noise to a previously JPEG-compressed image by searching for the intrinsic fingerprint of a specific pixel value mapping applied to the image in question.

[34] Key Concept – Digital contrast enhancement

Research – A wide variety of multimedia editing softwares, both commercial and open source, are currently available to every computer user. Content-preserving image manipulations such as resampling [4, 5], compression [6], contrast enhancement [7, 9], blurring [10], sharpening [11] and median filtering [12, 13] are detected or estimated passively [14]. In [7, 8], the blind forensic algorithms for detecting the globally and locally applied contrast enhancement have been proposed. In the recent paper [9], the authors propose an iterative algorithm to jointly estimate the contrast enhancement mapping used to modify an image and the pixel value histogram of the unenhanced image.

Methodology – To verify the validity of original contrast enhancement detection schemes [7, 8], the detection is performed on 346 images of the test set and their different contrast-enhanced versions. Detection results for traditional contrast enhancement operations are reported in figure. TP exceeds 0.99 when FP0≥0.07. Such results confirm the much reliable detection for a wide range of operation parameters. Figure in question may serve as reference for evaluating the capability of the proposed attacks to hide the traces of contrast enhancement operation.

Conclusion - As targeted anti-forensic techniques, two kinds of untraditional contrast enhancement methods have been proposed as targeted attacks against the state-of-the-art contrast enhancement forensic algorithms. The postprocessing type of attack is proposed by adding noise to the image enhanced by traditional contrast enhancement operations. Such an attack method can be applied in the case of unknowing the primary mapping function. The two proposed anti-forensic schemes are justified by plentiful experimental results on a large digital photograph image database for various parameter settings. Both undetectability and excellent visual quality have been manifested by comparing the test results with traditional contrast enhancement manipulations. The developed anti-forensic techniques examine the reliability of existing contrast enhancement forensic tools and could serve as targets for developing more secure forensic techniques.

[35] Key Concept – Contrast Enhancement Counter-Forensics

Research – Counter-forensics is a recent discipline aiming at gathering information about the history of images, video and audio contents, by relying on the subtle statistical traces left by every step in the life-cycle of a digital content. From an image forensic perspective, several approaches have been proposed so far to identify when a digital image undergone a contrast enhancement processing. The authors' goal is to reveal the presence of contrast enhancement by relying on second-order image statistics.

Methodology – The authors evaluate the performance of the proposed approach. The authors randomly selected 500 images from the UCID.v2 uncompressed color image dataset [17], of size pixels, and converted to greyscale, obtaining the Original Dataset. These images have been processed in two different ways: histogram stretching and gamma correction. The authors randomly selected 500 images from the UCID.v2 uncompressed color image dataset [17], of size pixels, and converted to greyscale, obtaining the Original Dataset. These images have been

processed in two different ways: histogram stretching and gamma correction. Gamma correction is applied with chosen in the set.

Conclusion - Counter-forensic techniques against detectors based on firstorder statistics have reached impressive results, making the use of such statistics irremediably insecure. Such counterforensic methods do not guarantee undetectability when higher statistics are employed. Motivated by this fact, the authors investigated the use of second-order statistics for detecting traces of histogram processing, devising a forensic algorithm based on the co-occurrence matrix. Experiments showed three interesting facts: i) the proposed detector slightly outperforms a reference first-order detector [4] even without attacks; ii) counter-forensic methods devised to fool first-order statistics detectors are not as effective against the second-order detector; iii) traces left by the considered attack in the second-order statistics allow to tell apart processed and attacked images.

[36] Key Concept – Adversary-Aware Double JPEG-Detection using Selected Training

Research – The authors' goal is to design an adversary-aware detector that reveals if an image has undergone a double JPEG compression possibly in the presence of other processing or counter-forensic attacks. The goal of this paper is to overcome the first order statistic limitation inherent in the analysis proposed in [10], and build an adversary-aware detector which is able to work under a wider variety of attacks. Image forensic techniques for double JPEG detection in the presence of an adversary is one of the most widely studied topics in adversarial image forensics [1], [2]. In real applications, the attack is not known in advance, impeding to build an ad-hoc detector. This problem is relevant with data driven detectors based on machine learning due the to difficulty of training the detector on all possible attacks.

Methodology – To perform the experiments the authors started with gray-scale images in uncompressed format. Part of them was used for training (Str) and part for testing (St). The authors built the images to be used for training and testing according to the following procedure: for the first class (H0), the images were single compressed with quality factor QF; for the second class (H1), the double compressed images were built by compressing the images first with various QF 1s and with QF For each image, the authors built a single compressed version with QF 2, many

double compressed versions with second quality factor QF 2, and the same number of attacked versions for each attack

Conclusion - The authors presented an adversary-aware double JPEG detector able to work even in the presence of heterogenous processing and C-F attacks. As regards the value of QF 2, the performance may decrease in case of mismatch between training and testing. To get good performance in this case, examples of images for which QF 1 > QF 2 must be included in the training set, even of the overall accuracy may decrease a bit. It is worth observing that the proposed approach is not meant for localization, and the performance are expected to decrease on small images. Exploring the link of the approach with one-class classifiers is another interesting research direction

[37] Key Concept – Manipulation Detection Using CNN

Research – Over the past several years, researchers have developed a variety of information forensic techniques to determine the authenticity and processing history of digital images. Recent experimental evidence has shown that tools initially developed to perform steganalysis are capable of detecting a wide variety of image editing operations. These tools from steganalysis operate by building local models of pixel dependencies by analyzing the joint distribution of pixel value prediction errors, extracting detection features from these joint distributions [16, 4]. These recent advances have been fueled by the use of GPUs to overcome the computational expense of estimating the large number of hyper-parameters that a deep network involves

Methodology – Median Filtering Gaussian Blurring AWGN, $\sigma = 2$ Re-sampling Accuracy. Each CNN corresponds to a binary classifier that detects one type of possible image operation with the same architecture outlined in Section 4. The authors chose 43, 500 unaltered blocks and their corresponding tampered blocks to build the training data for each type of forgery. The authors picked 16, 000 unaltered blocks and their corresponding tampered blocks to build the testing data for each type of forgery. For every binary classifier the authors have a total number of 87, 000 blocks for training and 32, 000 blocks for testing

Conclusion - The authors proposed a novel CNN-based universal forgery detection technique that can automatically learn how to detect different image manipulations. Through a series of experiments, the authors demonstrated that the CNN-based universal forensic approach can automatically learn how to detect multiple image manipulations without relying on pre-selected

features or any preprocessing. The results of these experiments demonstrated that the proposed approach can automatically detect several different manipulations with an average accuracy of 99.10%

[38] Key Concept – Large Scale Image Recognition

Research – Convolutional networks (ConvNets) have recently enjoyed a great success in large-scale image and video recognition (Krizhevsky et al, 2012; Zeiler & Fergus, 2013; Sermanet et al, 2014; Simonyan & Zisserman, 2014) which has become possible due to the large public image repositories, such as ImageNet (Deng et al, 2009), and high-performance computing systems, such as GPUs or large-scale distributed clusters (Dean et al, 2012). The best-performing submissions to the ILSVRC2013 (Zeiler & Fergus, 2013; Sermanet et al, 2014) utilised smaller receptive window size and smaller stride of the first convolutional layer Another line of improvements dealt with training and testing the networks densely over the whole image and over multiple scales (Sermanet et al, 2014; Howard, 2014). The authors fix other parameters of the architecture, and steadily increase the depth of the network by adding more convolutional layers, which is feasible due to the use of very small (3 × 3) convolution filters in all layers

Methodology – The authors considered an ensemble of only two best-performing multi-scale models, which reduced the test error to 7.0% using dense evaluation and 6.8% using combined dense and multi-crop evaluation. The authors' best-performing single model achieves 7.1% error. As can be seen from Table 7, the very deep ConvNets significantly outperform the previous generation of models, which achieved the best results in the ILSVRC-2012 and ILSVRC-2013 competitions. In terms of the single-net performance, the architecture achieves the best result (7.0% test error), outperforming a single GoogLeNet by 0.9%.

Conclusion - The authors' ConvNet configurations are quite different from the ones used in the top-performing entries of the ILSVRC-2012 (Krizhevsky et al, 2012) and ILSVRC-2013 competitions (Zeiler & Fergus, 2013; Sermanet et al, 2014). Layers (e.g. 11 × 11 with stride 4 in (Krizhevsky et al, 2012), or 7 × 7 with stride 2 in (Zeiler & Fergus, 2013; Sermanet et al, 2014)), the authors use very small 3 × 3 receptive fields throughout the whole net, which are convolved with the input at every pixel. The authors' results yet again confirm the importance of depth in visual representations.

[39] Key Concept – High Resolution Image Dataset mainly meant for forensic algorithms.

Research – Digital forensics is a relatively new research area which aims at authenticating digital media by detecting possible digital forgeries. Indeed, the ever-increasing availability of multimedia data on the web, coupled with the great advances reached by computer graphical tools, makes the modification of an image and the creation of visually compelling forgeries an easy task for any user. This in turns creates the need of reliable tools to validate the trustworthiness of the represented information. In such a context, we present here RAISE, a large dataset of 8156 high-resolution raw images, depicting various subjects and scenarios, properly annotated and available together with accompanying metadata. Such a wide collection of untouched and diverse data is intended to become a powerful resource for, but not limited to, forensic researchers by providing a common benchmark for a fair comparison, testing and evaluation of existing and next generation forensic algorithms. In this paper we describe how RAISE has been collected and organized, discuss how digital image forensics and many other multimedia research areas may benefit of this new publicly available benchmark dataset and test a very recent forensic technique for JPEG compression detection.

Methodology – Digital forensics is a relatively new research area which aims at authenticating digital media by detecting possible digital forgeries. The ever-increasing availability of multimedia data on the web, coupled with the great advances reached by computer graphical tools, makes the modification of an image and the creation of visually compelling forgeries an easy task for any user. This in turns creates the need of reliable tools to validate the trustworthiness of the represented information. The authors present here RAISE, a large dataset of 8156 high-resolution raw images, depicting various subjects and scenarios, properly annotated and available together with accompanying metadata.

Conclusion - Such a wide collection of untouched and diverse data is intended to become a powerful resource for, but not limited to, forensic researchers by providing a common benchmark for a fair comparison, testing and evaluation of existing and generation forensic algorithms. In this paper the authors describe how RAISE has been collected and organized, discuss how digital image forensics and many other multimedia research areas may benefit of this new publicly available benchmark dataset and test a very recent forensic technique for JPEG compression detection.

[40] Key Concept – Large Scale Image Recognition

Research – Convolutional networks (ConvNets) have recently enjoyed a great success in large-scale image and video recognition (Krizhevsky et al, 2012; Zeiler & Fergus, 2013; Sermanet et al, 2014; Simonyan & Zisserman, 2014) which has become possible due to the large public image repositories, such as ImageNet (Deng et al, 2009), and high-performance computing systems, such as GPUs or large-scale distributed clusters (Dean et al, 2012). The best-performing submissions to the ILSVRC2013 (Zeiler & Fergus, 2013; Sermanet et al, 2014) utilised smaller receptive window size and smaller stride of the first convolutional layer Another line of improvements dealt with training and testing the networks densely over the whole image and over multiple scales (Sermanet et al, 2014; Howard, 2014). The authors fix other parameters of the architecture, and steadily increase the depth of the network by adding more convolutional layers, which is feasible due to the use of very small (3 × 3) convolution filters in all layers

Methodology – The authors considered an ensemble of only two best-performing multi-scale models, which reduced the test error to 7.0% using dense evaluation and 6.8% using combined dense and multi-crop evaluation. The authors' best-performing single model achieves 7.1% error. As can be seen from Table 7, the very deep ConvNets significantly outperform the previous generation of models, which achieved the best results in the ILSVRC-2012 and ILSVRC-2013 competitions. In terms of the single-net performance, the architecture achieves the best result (7.0% test error), outperforming a single GoogLeNet by 0.9%.

Conclusion - The authors' ConvNet configurations are quite different from the ones used in the top-performing entries of the ILSVRC-2012 (Krizhevsky et al, 2012) and ILSVRC-2013 competitions (Zeiler & Fergus, 2013; Sermanet et al, 2014). Layers (e.g. 11 × 11 with stride 4 in (Krizhevsky et al, 2012), or 7 × 7 with stride 2 in (Zeiler & Fergus, 2013; Sermanet et al, 2014)), the authors use very small 3 × 3 receptive fields throughout the whole net, which are convolved with the input at every pixel. The authors' results yet again confirm the importance of depth in visual representations.

Preliminaries

The working environment would be Google Colaboratory for inherent GPU access and cloud storage. The language for modeling our auto-encoder would be python3. The various python3 libraries included in the project to obtain necessary outputs are as follows:

1. **NumPy** – A general purpose array-processing package. It provides high-performance multidimensional array objects and tools supporting operations on these arrays. It is the fundamental package for scientific computing with Python.
2. **Matplotlib** – It is a plotting library for Python and its numerical mathematics extension

NumPy

3. **OS** – The OS module provides functions for interacting with the underlying operating system and its dependent functionality. It comes under Python's standard utility modules.
4. **cv2/OpenCV** – It is a library of Python bindings designed to solve computer vision problems. All the underlying OpenCV array structures are converted to and from NumPy arrays. It also makes integration with other libraries much easier such as SciPy and Matplotlib.
5. **Keras with Tensorflow backend** – It's open-source neural-network library which can run on top of a series of other libraries and software such as Theano, Tensorflow, PlaidML etc.

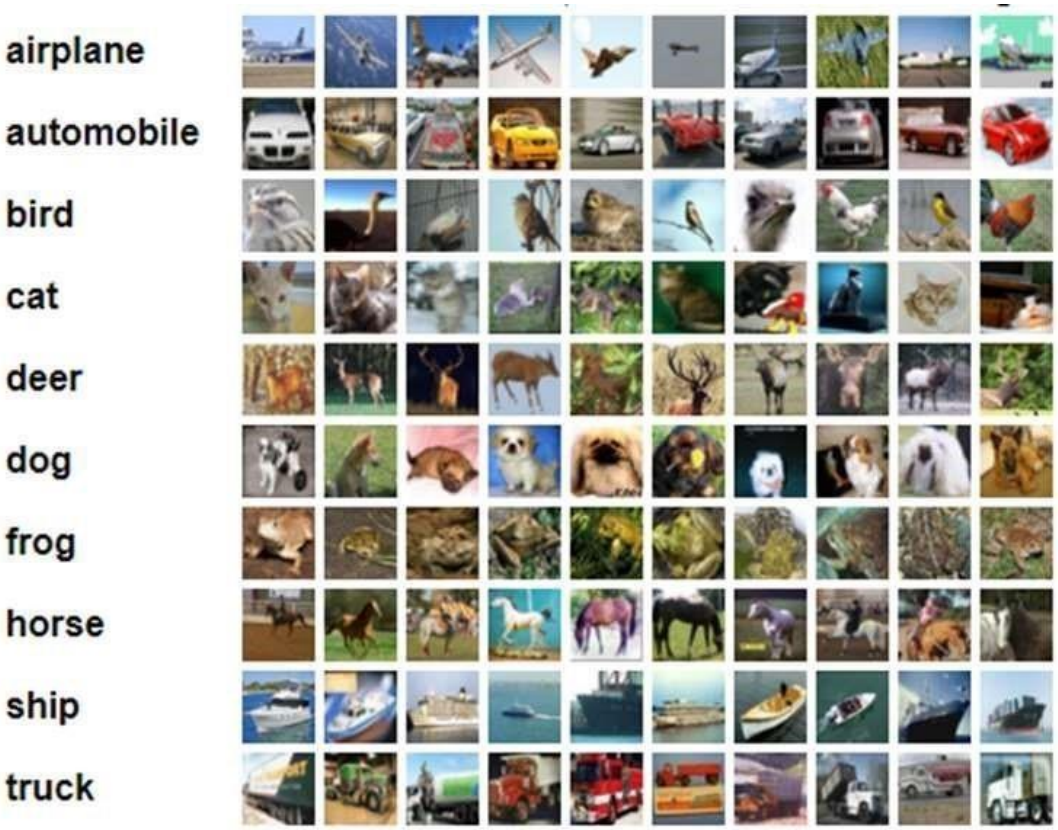
Neural Network Components

1. **Layers** – It refers to the neural network layers, there are different layers used in an autoencoder model.
    1. **Dense** – It builds layers such that each node in an input layer is connected to the output layer.
    2. **Conv2D** – It develops kernels that extract features from the images
    3. **Flatten** – Converts the 2D matrix into 1D matrix. Reshapes the matrix layers to a list
    4. **Conv2DTranspose** – It is the transpose operation of Conv2D focused on building back the image from the filter aggregation.
    5.
2. **Model** – The object wrapped around the layer

Dataset

The CIFAR-10 dataset is one of the most diverse and rich data sets for image training models. It consists of 60,000 32×32 color images with 10 classes, computing over to 6000 images per class. The model would be using 50,000 training images and 10,000 test images. The data set is divided into five training batches and one test batch, each with 10,000 images. The test batch consists of exactly 1,000 randomly selected images form each class. The training set contains exactly 5,000 images for each class. As the CIFAR-10 data set is coloured, we proceed with manually converting our training data set into Black and white through the help of computer vision tools like OpenCV. We will then proceed to use them in the training process.

All CIFAR-10 classes with accompanying sample images[30]



Review 2 (Design of Diagrams & Prototype Design)

Proposed Methodology

Input/Output

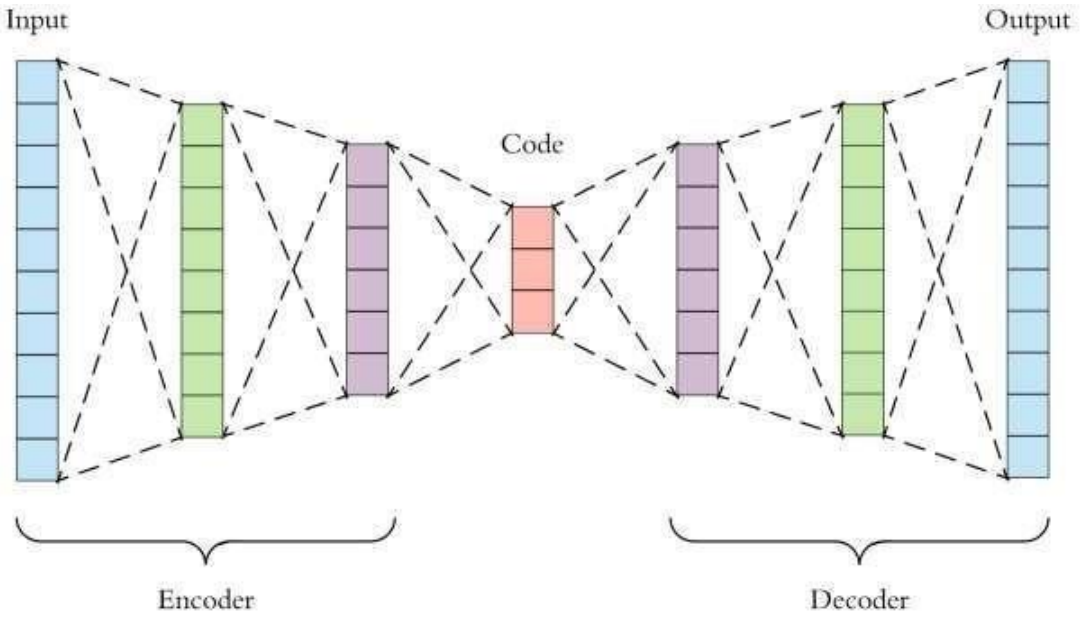
Our approach consists of training a CNN auto-encoder model with Black-and-white images and allowing it to learn coloured features without human supervision. To be more specific, we will convert the given coloured images into single-channel Black and white color space images that can provide us a grayscale input to our model. We then pass this model to our auto-encoder and it passes through the encoder schema first. The encoder will apply the convolution operation multiple times and return a vector of embeddings. These embeddings would be then passed to our decoder model which will apply the transpose convolution operation and return an output of the same height and width but with 3 different channels – RGB. The image dimensions and other properties would be retained throughout.

Input Image size – 32×32 pixel, single channel black and white images

Output Image size – 32×32 pixel, three channel image (RGB)

Encoding generation – [256] embeddings per image generated in a latent vector

Schematic representation of a general auto-encoder model for data compression[32]



Algorithm – Architecture and Flow

The model consists of an auto-encoder model which is a horizontal stack of the encoder and decoder models. The black and white image would first flow through the encoder model to be aggregated in a vector of embeddings. The process is as follows:

1. The image is passed to the encoder model
    1. The image undergoes convolution by 64 filters (dimension – 3×3) with a stride of 2, no padding and ReLU activation. It results in dimensions – 16×16×64
    2. The image undergoes convolution by 128 filters (dimension – 3×3) with a stride of 2, no padding and ReLU activation. It results in dimensions – 8×8×128
    3. The image undergoes convolution by 256 filters (dimension – 3×3) with a stride of 2, no padding and ReLU activation. It results in dimensions – 4×4×256
    4. The resultant vector is spread out into a 1D vector (dimension – [256]).
2. The embeddings vector (latent_vector) is passed to the decoder model to apply the transpose convolution operation:
    1. A dense neural network layer is applied onto the latent vector. It results in a 1D vector (dimension [4096])
    2. The vector is reshaped into a three dimensional vector – (4×4×256)
    3. The image undergoes convolution by 256 filters (dimension – 3×3) with a stride of 2, no padding and ReLU activation. It results in dimensions – 8×8×256
    4. The image undergoes convolution by 128 filters (dimension – 3×3) with a stride of 2, no padding and ReLU activation. It results in dimensions – 16×16×128
    5. The image undergoes convolution by 64 filters (dimension – 3×3) with a stride of 2, no padding and ReLU activation. It results in dimensions – 32×32×64
    6. The image undergoes convolution by 3 filters (dimension – 3×3) with a stride of 2, no padding and sigmoid activation. It results in dimensions – 32×32×3

The resulting image from the encoder model is displayed as the output image. We successfully retain the height and width of the image (32×32) throughout the colorization process.

1. The architecture of the image passed in step 1 (the encoder model):

Layer	**Output Shape**	**Trainable Parameters**
encoder_input (Input Layer)	(None, 32, 32, 1)	0
Conv2d_1 (Conv2D)	(None, 16, 16, 64)	640
Conv2d_2 (Conv2D)	(None, 8, 8, 128)	73856
Conv2d_3 (Conv2D)	(None, 4, 4, 256)	295168
Flatten_1 (Flatten)	(None, 4096)	0
Latent_vector (Dense)	(None, 256)	1048832

2. The architecture of the image passed in step 2 (the decoder model):

Layer	**Output Shape**	**Trainable Parameters**
decoder_input (Input Layer)	(None, 256)	0
dense_1 (Dense)	(None, 4096)	1052672
Reshape_1 (Reshape)	(None, 4, 4, 256)	0
Conv2d_transpose_1 (Conv2DTranspose)	(None, 8, 8, 256)	590080
Conv2d_transpose_2 (Conv2DTranspose)	(None, 16, 16, 128)	295040
Conv2d_transpose_3 (Conv2DTranspose)	(None, 32, 32, 64)	73792
decoder_output (Conv2DTranspose)	(None, 32, 32, 3)	1731

The final aggregated model:

Layer	Output Shape	Trainable Parameters
encoder_input (Input Layer)	(None, 32, 32, 1)	0
encoder_model (Model)	(None, 4096)	1418496
decoder_model (Model)	(None, 32, 32, 3)	2013315

Data Preprocessing

Processing images to single channel black-and-white

Since there is no proper image dataset for the tasks of automatic colorization, we will be using the processed CIFAR-10 dataset where each and every image would be converted into black and white. The black-and-white single channel images would be the features for our training data while the coloured image would be the target.

Normalization

An 8 bit image is represented by 255 shades of colours i.e. pixel values. A broad range of pixel values can hurt the training of a model and cause it to over-fit and take much longer to learn parameter values. In order to prevent unwanted interference, the image should be normalized by dividing it with 255 so that the final values lie between the range of [0-1]. Each pixel here will be given equal preference

Activation Function

ReLU stands for rectified linear unit and it is the main activation function used in the hidden layers of our model. It is defined by the following mathematical equation.

$$y = \max(0, x)$$

Learning Rate

The learning rate is scheduled with the plateau function. After a set number of epochs, if the model does not show any improvement in validation accuracy, it will divide the learning rate by a set value. There is also a lower limit set to the learning rate decliner function so that the learning rate doesn't fall below a certain value. It may give rise to problems such as vanishing gradients and would also result in the model taking up too much time to train effectively.

Optimizer

The Adam optimizer [26] is one of the optimizers experimented with. Adam stands for **Adaptive moment estimation** combines the ideas of momentum optimization and RMSProp: just like momentum optimization, it keeps track of an exponentially decaying average of past gradients; and just like RMSProp, it keeps track of an exponentially decaying average of past squared gradients.

Here, Beta1 refers to the momentum decay hyperparameter while Beta2 refers to the scaling decay hyperparameter. The epsilon parameter is the smoothing terms which is usually initialized to (10^{-7}).

As Adam is an adaptive learning rate algorithm like AdaGrad and RMSProp. It requires less tuning of the learning rate hyperparameter (n). The default value n=0.001 works for most cases, making it easier to use than gradient descent.

1. $\mathbf{m} \leftarrow \beta_1 \mathbf{m} - (1 - \beta_1)\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
2. $\mathbf{s} \leftarrow \beta_2 \mathbf{s} + (1 - \beta_2)\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \otimes \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
3. $\widehat{\mathbf{m}} \leftarrow \dfrac{\mathbf{m}}{1 - \beta_1^{\,t}}$
4. $\widehat{\mathbf{s}} \leftarrow \dfrac{\mathbf{s}}{1 - \beta_2^{\,t}}$
5. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \widehat{\mathbf{m}} \oslash \sqrt{\widehat{\mathbf{s}} + \varepsilon}$

The Adam Algorithm for updating parameters (Theta) [26]

The default gradient descent algorithm which is the mini-batch gradient descent or the Stochastic Gradient descent is almost always outperformed by other gradient descent algorithms. Adam optimization has two sub-categories – Nadam and AdaMax. These are for more specific cases so we will be using the parent optimization algorithm.

Another optimizer we will be looking at is RMSProp which is another versatile gradient descent algorithm.

The **RMSProp algorithm [33]** fixes other gradient descent algorithms such as AdaGrad. RMSProp has an interesting background in that it was introduced by Geoff Hinton on Coursera first and was spread widely after. AdaGrad runs the risk of slowing down a bit too fast and never converges to the global optimum. It accumulates gradients from the most recent iterations (instead of all gradients since the beginning of training). It scales down the gradient vectors along the steepest dimension. It does so by the process of exponential decay.

$$1. \quad \mathbf{s} \leftarrow \beta \mathbf{s} + (1 - \beta) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \otimes \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$2. \quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \oslash \sqrt{\mathbf{s} + \varepsilon}$$

The decay rate is Beta and it is typically set to the value of 0.9 (default value). It is another hyperparameter but the default value works for most cases. This optimizer usually works better than AdaGrad. It was the most preferred algorithm for a long time until Adam optimization came along and worked better than RMSProp in some cases.

Dropout

Dropout is an amazingly compelling regularization method presented by Srivastava et al. [27]. Dropout is a method where essentially each neuron has a probability to be put to sleep during the training step/epoch.

It may be active during the next phase of training. We add another hyperparameter p which is the dropout rate – the probability of a neuron to be put to sleep during the next epoch. Neurons trained with dropout cannot adapt with their neighboring neurons, they cannot rely excessively on just a few neurons and their weights. They end up being less sensitive to slight changes in the input.

By the end of the dropout process, the neural network is much more robust. A unique model is

generated at each training step. The neural networks are not independent because they share many weights but the resulting network can be seen as an averaging ensemble of all the smaller neural network.

Here, we are working with the dropout rate of 0.2 for all convolution layers

Objective Function

As brought up in [20, 7, 8], one of the most significant difficulties in auto-colorization is a cost function that represents the multi-modal idea of the issue. To explore this further, loss functions were tested.

The L1 Loss function can be represented in the following method. The parameters are the same as above. It was was established by Girshick [28].

$$\frac{1}{n}\sum_{n}\begin{pmatrix}0.5\times(x_i-y_i)^2\ if\ |x_i-y_i|<1\\ |x_i-y_i|-0.5\ otherwise\end{pmatrix}$$

This loss penalizes outliers less and it can result in more blurry images as the pixel values could be completely inaccurately predicted. This could lead to the predicted value being far away from the ground truth value and can result in wrong coloring. We observe that this function would be much more harmful than the L2 Loss function.

The L2 Loss function: x_i represents the predicted pixel values and y_i the ground truth value. N is the total number of pixels across all input images to the model.

$$\frac{1}{n}\sum_{n}(x_i-y_i)^2$$

This is our primary loss function and we would be using this as our means for the final prediction accuracy and validation loss functions. This loss is also known as the MSE (mean squared error) or the quadratic loss function. This loss metric is not robust to outliers and penalizes them hard enough to capture the finer details in the final predictions made by our auto-encoder model.

We will also experiment with the cross_entropy loss function to compare the predicted image colorization, hues and saturation levels to cross-verify and examine the best loss metric for our current task of autocolorization.

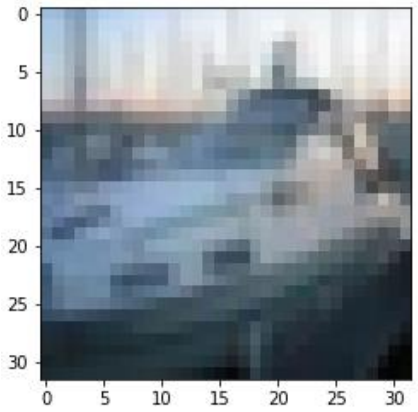
Review 3 (Prototyping and Development of Model)

The model development underwent several stages before arriving at a single suitable and stable model that provided us with vibrant and natural colour images. The model was fine-tuned and experimented with different optimizers and regularization strategies. Some of the major areas the model development focused on was:

- Dropout Rates
- Batch Normalization of Convolution layers
- Optimizer selection
- Loss metric selection

Through extensive tuning and predictive results, the optimizer and loss metric was evaluated. Our model was best fit for the **mean squared error loss** metric. The optimizers performed at similar capabilities with key ones such as **RMSProp, Nadam & Adam** performing near to the same. We decided to proceed with **Adam** as the key metric optimizer due to its adaptive nature through which the learning rate would auto-tune and be learned just like the other layer parameters.

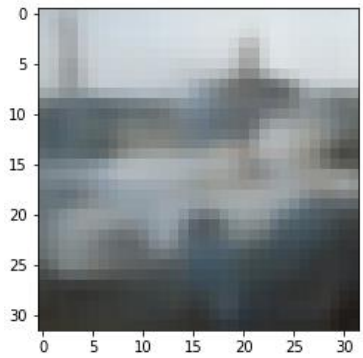
Extensive testing was done to decide upon the dropout and batch normalization. The results were recorded and analysed. A reference coloured image was used as a visual comparator for different model results.



Our base image for reference

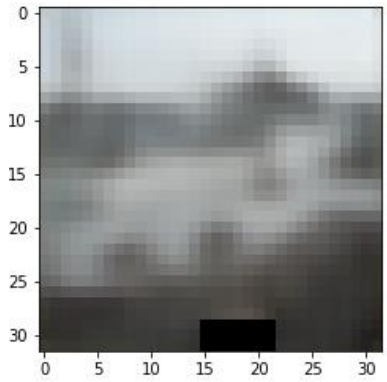
Experiments with different regularization techniques was undertaken with the strategy of obtaining the most vibrant image which resembles our reference image the closest:

- Using Batch Normalization as well as a dropout of about 20% resulted in validation accuracies of 0.52 (not to compare as drop-outs consist of neurons shutting down during training). Our final predicted image was very normalized and each pixel was about the mean pixel value:



Using Batch Normalization + Dropout

- Proceeding with normal or high levels of drop-outs only without batch normalization resulted in high normalization of pixel values along the mean pixel colour. This gave us a validation accuracy of 0.52 as well but the predicted images were further desaturated



Using high level of dropouts

- Proceeding without regularization gave us the best results (even though it is counter intuitive to the principle of regularization). Our images were far more vibrant and dynamic in saturation levels. The validation accuracy was a bit low (0.509) but it still produced the best images

We therefore proceeded with the

Regular CNN implementation without regularization and the code is as follows:

Code implementation

```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
from keras.layers import Dense, Input, Conv2D, Flatten, Reshape, Conv2DTra
nspose, Dropout
from keras.models import Model
from tensorflow.keras.layers import BatchNormalization
from keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
from keras.datasets import cifar10
from keras.utils import plot_model
from keras import backend as K
```

```
#Using cv2.cvtcolor(image, cv2.color_scheme)
#Input coloured image and returned the gray image
def rgb_gray(img):
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return gray_img


#Loading the training and testing dataset from Keras datasets
#Here, x_train is our training set and x_test is our validation set
#We do not require the y_train/y_test dataset as we are not performing ima
ge classification with target labels
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
#Dimensions of each of the sets
print(x_train.shape)
print(x_test.shape)
#Getting the height, width and number of channels from the image
img_dim = x_train.shape[1]
channels = 3


#Display sample images from the dataset
fig, ax = plt.subplots(5,5,sharex=True,sharey=True,figsize=(10,10))
c=0
for i in range(0,5):
    for j in range (0,5):
        ax[i,j].imshow(x_train[c])
        ax[i,j].set_title(y_train[c])
        c=c+1


#Our lists to store the converted images
x_train_Gray = []
x_test_Gray = []
#Iterate over number of images for training set
for i in range(x_train.shape[0]):
 img = x_train[i]
 #convert and append
 x_train_Gray.append(rgb_gray(img))
print(len(x_train_Gray))


#Same process for the test images
for i in range(x_test.shape[0]):
 img = x_test[i]
 x_test_Gray.append(rgb_gray(img))
print(len(x_test_Gray))
```

```
#Convert the given list to a numpy array for input
x_train_Gray = np.asarray(x_train_Gray)
x_test_Gray = np.asarray(x_test_Gray)

#Reshape the given data into (m, height, width, channels)
#We will reshape both the training coloured and gray images - Features and
 targets
x_train = x_train.reshape(x_train.shape[0], img_dim, img_dim, channels)
x_test = x_test.reshape(x_test.shape[0], img_dim, img_dim, channels)
x_train_Gray = x_train_Gray.reshape(x_train_Gray.shape[0], img_dim,
img_dim, 1)
x_test_Gray = x_test_Gray.reshape(x_test_Gray.shape[0], img_dim,img_dim,1)
input_shape = (img_dim, img_dim, 1)
lat_dim = 256


#Dividing each pixel by 255 to get a normalized value
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255
x_train_Gray = x_train_Gray.astype('float32')/255
x_test_Gray = x_test_Gray.astype('float32')/255


im1 = x_train[0]
fig, ax = plt.subplots(ncols = 2)
ax[0].imshow(im1)
im2 = cv2.cvtColor(im1, cv2.COLOR_RGB2GRAY)
ax[1].imshow(im2, cmap='gray')


#ENCODER MODEL
inputs = Input(shape=input_shape, name='encoder_input')
x = inputs
x = Conv2D(64, (3, 3), strides=2, activation='relu', padding='same')(x)
x = Conv2D(128, (3, 3), strides=2, activation='relu', padding='same')(x)
x = Conv2D(256, (3, 3), strides=2, activation='relu', padding='same')(x)
shape = K.int_shape(x)
x = Flatten()(x)
latent = Dense(lat_dim, name='latent_vector')(x)
encoder = Model(inputs, latent, name='encoder_model')
encoder.summary()


#DECODER MODEL
latent_inputs = Input(shape = (lat_dim,), name = 'decoder_input')
x = Dense(shape[1]*shape[2]*shape[3])(latent_inputs)
x = Reshape((shape[1], shape[2], shape[3]))(x)
x = Conv2DTranspose(256, (3,3), strides = 2, activation = 'relu', padding=
 'same')(x)
```

```
x = Conv2DTranspose(128, (3,3), strides = 2, activation = 'relu', padding=
 'same')(x)
x = Conv2DTranspose(64, (3,3), strides = 2, activation = 'relu', padding =
'same')(x)
outputs = Conv2DTranspose(3, (3,3), activation = 'sigmoid', padding = 'sam
e', name = 'decoder_output')(x)
decoder = Model(latent_inputs, outputs, name = 'decoder_model')
decoder.summary()


#AUTOENCODER MODEL
autoencoder = Model(inputs, decoder(encoder(inputs)), name = 'autoencoder'
)
autoencoder.summary()


#Learning Rate plateau and scheduling
lr_reducer = ReduceLROnPlateau(factor = np.sqrt(0.1), cooldown = 0, patien
ce = 5, verbose = 1, min_lr = 0.5e-6)
#Preparing save location during model training
save_dir = os.path.join(os.getcwd(), 'saved_models')
model_name = 'colorized_ae_model.h5'

#Creating dir to save the model
if not os.path.isdir(save_dir):
  os.makedirs(save_dir)
filepath = os.path.join(save_dir, model_name)
print(filepath)

#Checkpoint and learning rate scheduling callback
checkpoints = ModelCheckpoint(filepath = filepath, monitor = 'val_loss', v
erbose = 1, save_best_only = True)
callbacks = [lr_reducer, checkpoints]


#TRAINING
#Using the desired loss function, optimizer and metric
autoencoder.compile(loss = "mse", optimizer = 'Adam', metrics = ['accuracy
'])
#Fit the model with the training set and validation set
autoencoder.fit(x_train_Gray, x_train, validation_data = (x_test_Gray, x_t
est),epochs = 50, batch_size = 32, callbacks = callbacks)
x_decoded = autoencoder.predict(x_test_Gray)
autoencoder.save('colourization_model.h5')
```

```
from google.colab import files
#files.download('colourization_model.h5')
#Displaying Results
imgs = x_test[:25]
imgs = imgs.reshape((5, 5, img_dim, img_dim, channels))
imgs = np.vstack([np.hstack(i) for i in imgs])
plt.figure()
plt.axis('off')
plt.title('Original Image')
plt.imshow(imgs, interpolation='none')
#plt.savefig('%s/colorized.png' % imgs_dir)
plt.show()
imgs = x_decoded[:25]
imgs = imgs.reshape((5, 5, img_dim, img_dim, channels))
imgs = np.vstack([np.hstack(i) for i in imgs])
plt.figure()
plt.axis('off')
plt.title('Colorized test images (Predicted)')
plt.imshow(imgs, interpolation='none')
#plt.savefig('%s/colorized.png' % imgs_dir)
plt.show()


df = pd.DataFrame(history.history)
df[['accuracy','val_accuracy']].plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1) # set the vertical range to [0-1]
plt.xlabel("epochs")
plt.title('Train and Validation Accuracy')
plt.show()


df[['val_loss']].plot()
plt.xlabel("epochs")
plt.title('Validation Loss')
plt.show()


df[['loss']].plot()
plt.xlabel("epochs")
plt.title('Training Loss')
plt.show()
```

Screenshots of code execution

Autoencoder model for auto-colorization

Library imports

```
[ ] import numpy as np
    import matplotlib.pyplot as plt
    import os
    import cv2
    from keras.layers import Dense, Input, Conv2D, Flatten, Reshape, Conv2DTranspose, Dropout
    from keras.models import Model
    from tensorflow.keras.layers import BatchNormalization
    from keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
    from keras.datasets import cifar10
    from keras.utils import plot_model
    from keras import backend as K
```

Defining the RGB to Gray Transition function

```
[ ] #Using cv2.cvtcolor(image, cv2.color_scheme)
    #Input coloured image and returned the gray image
    def rgb_gray(img):
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        return gray_img
```

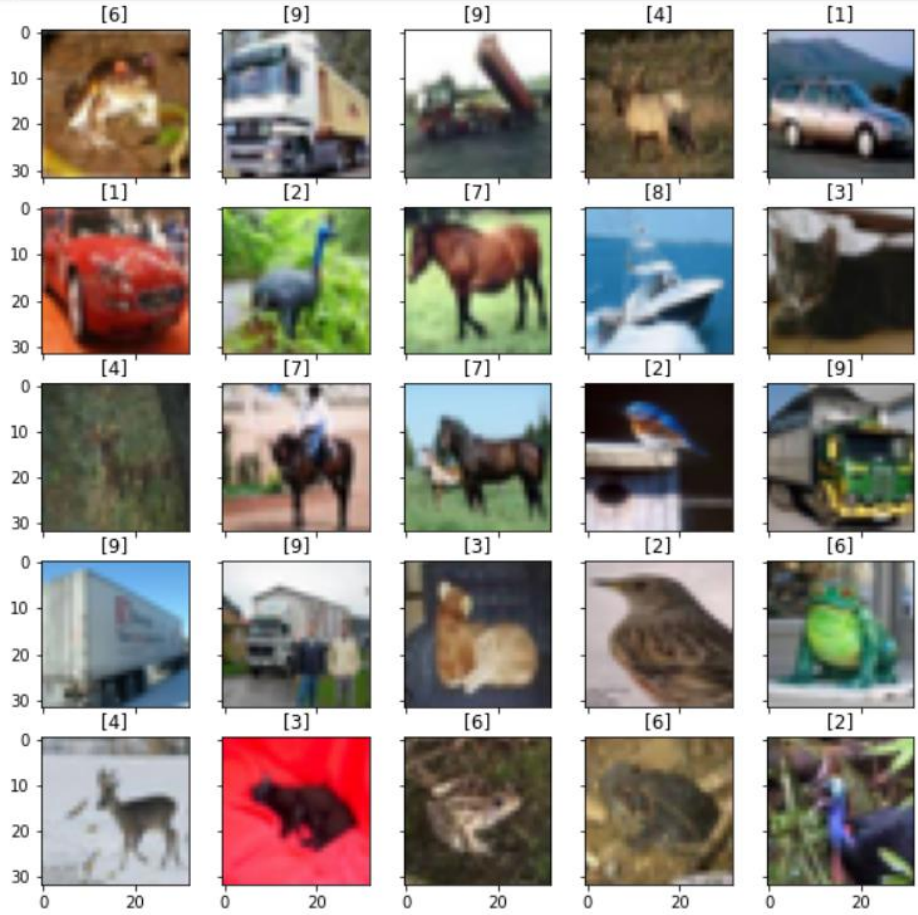
Loading Data

```
[ ] #Loading the training and testing dataset from Keras datasets
    #Here, x_train is our training set and x_test is our validation set
    #We do not require the y_train/y_test dataset as we are not performing image classification with target labels
    (x_train, y_train), (x_test, y_test) = cifar10.load_data()
    #Dimensions of each of the sets
    print(x_train.shape)
    print(x_test.shape)
    #Getting the height, width and number of channels from the image
    img_dim = x_train.shape[1]
    channels = 3
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 6s 0us/step
(50000, 32, 32, 3)
(10000, 32, 32, 3)
```

Displaying sample images from the dataset with their respective class

```
#Display sample images from the dataset
fig, ax = plt.subplots(5,5,sharex=True,sharey=True,figsize=(10,10))
c=0
for i in range(0,5):
    for j in range (0,5):
        ax[i,j].imshow(x_train[c])
        ax[i,j].set_title(y_train[c])
        c=c+1
```



Conversion process and storing them in a list

```
[ ] #Our lists to store the converted images
    x_train_Gray = []
    x_test_Gray = []
    #Iterate over number of images for training set
    for i in range(x_train.shape[0]):
     img = x_train[i]
     #convert and append
     x_train_Gray.append(rgb_gray(img))
    print(len(x_train_Gray))
```

```
50000
```

```
[ ] #Same process for the test images
    for i in range(x_test.shape[0]):
     img = x_test[i]
     x_test_Gray.append(rgb_gray(img))
    print(len(x_test_Gray))
```

```
10000
```

Image Pre-processing

```
[ ] #Convert the given list to a numpy array for input
    x_train_Gray = np.asarray(x_train_Gray)
    x_test_Gray = np.asarray(x_test_Gray)

    #Reshape the given data into (m, height, width, channels)
    #We will reshape both the training coloured and gray images - Features and targets
    x_train = x_train.reshape(x_train.shape[0], img_dim, img_dim, channels)
    x_test = x_test.reshape(x_test.shape[0], img_dim, img_dim, channels)
    x_train_Gray = x_train_Gray.reshape(x_train_Gray.shape[0], img_dim,
    img_dim, 1)
    x_test_Gray = x_test_Gray.reshape(x_test_Gray.shape[0], img_dim,img_dim,1)
    input_shape = (img_dim, img_dim, 1)
    lat_dim = 256
```

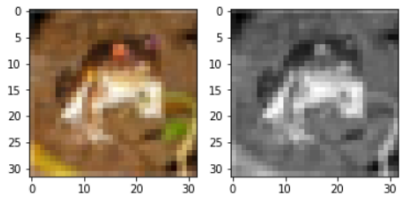
Normalizing pixels over the range [0-1]

```
[ ] #Dividing each pixel by 255 to get a normalized value
    x_train = x_train.astype('float32')/255
    x_test = x_test.astype('float32')/255
    x_train_Gray = x_train_Gray.astype('float32')/255
    x_test_Gray = x_test_Gray.astype('float32')/255
```

Example of sample image and converted image (after passing through black and white function)

```
[ ] im1 = x_train[0]
    fig, ax = plt.subplots(ncols = 2)
    ax[0].imshow(im1)
    im2 = cv2.cvtColor(im1, cv2.COLOR_RGB2GRAY)
    ax[1].imshow(im2, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f8336bf9278>
```



Initialization of the encoder model

```
[ ] #ENCODER MODEL
    inputs = Input(shape=input_shape, name='encoder_input')
    x = inputs
    x = Conv2D(64, (3, 3), strides=2, activation='relu', padding='same')(x)
    x = Conv2D(128, (3, 3), strides=2, activation='relu', padding='same')(x)
    x = Conv2D(256, (3, 3), strides=2, activation='relu', padding='same')(x)
    shape = K.int_shape(x)
    x = Flatten()(x)
    latent = Dense(lat_dim, name='latent_vector')(x)
    encoder = Model(inputs, latent, name='encoder_model')
    encoder.summary()
```

```
Model: "encoder_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   [(None, 32, 32, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 16, 16, 64)        640
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 256)         295168
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
latent_vector (Dense)        (None, 256)               1048832
=================================================================
Total params: 1,418,496
Trainable params: 1,418,496
Non-trainable params: 0
```

Initialization of the decoder model

```
#DECODER MODEL
latent_inputs = Input(shape = (lat_dim,), name = 'decoder_input')
x = Dense(shape[1]*shape[2]*shape[3])(latent_inputs)
x = Reshape((shape[1], shape[2], shape[3]))(x)
x = Conv2DTranspose(256, (3,3), strides = 2, activation = 'relu', padding= 'same')(x)
x = Conv2DTranspose(128, (3,3), strides = 2, activation = 'relu', padding= 'same')(x)
x = Conv2DTranspose(64, (3,3), strides = 2, activation = 'relu', padding ='same')(x)
outputs = Conv2DTranspose(3, (3,3), activation = 'sigmoid', padding = 'same', name = 'decoder_output')(x)
decoder = Model(latent_inputs, outputs, name = 'decoder_model')
decoder.summary()
```

```
Model: "decoder_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
decoder_input (InputLayer)   [(None, 256)]             0
_________________________________________________________________
dense (Dense)                (None, 4096)              1052672
_________________________________________________________________
reshape (Reshape)            (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 8, 8, 256)         590080
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 16, 16, 128)       295040
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 32, 32, 64)        73792
_________________________________________________________________
decoder_output (Conv2DTransp (None, 32, 32, 3)         1731
=================================================================
Total params: 2,013,315
Trainable params: 2,013,315
```

Auto-encoder model

```
[ ] #AUTOENCODER MODEL
    autoencoder = Model(inputs, decoder(encoder(inputs)), name = 'autoencoder')
    autoencoder.summary()
```

```
Model: "autoencoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   [(None, 32, 32, 1)]       0
_________________________________________________________________
encoder_model (Functional)   (None, 256)               1418496
_________________________________________________________________
decoder_model (Functional)   (None, 32, 32, 3)         2013315
=================================================================
Total params: 3,431,811
Trainable params: 3,431,811
Non-trainable params: 0
_________________________________________________________________
```

Learning rate scheduling and checkpoint callbacks

```
[ ] #Learning Rate plateau and scheduling
    lr_reducer = ReduceLROnPlateau(factor = np.sqrt(0.1), cooldown = 0, patience = 5, verbose = 1, min_lr = 0.5e-6)
    #Preparing save location during model training
    save_dir = os.path.join(os.getcwd(), 'saved_models')
    model_name = 'colorized_ae_model.h5'

    #Creating dir to save the model
    if not os.path.isdir(save_dir):
      os.makedirs(save_dir)
    filepath = os.path.join(save_dir, model_name)
    print(filepath)

    #Checkpoint and learning rate scheduling callback
    checkpoints = ModelCheckpoint(filepath = filepath, monitor = 'val_loss', verbose = 1, save_best_only = True)
    callbacks = [lr_reducer, checkpoints]
```

```
/content/saved_models/colorized_ae_model.h5
```

Training the model over the given dataset over batches and epochs

```
#TRAINING
#Using the desired loss function, optimizer and metric
autoencoder.compile(loss = "mse", optimizer = 'Adam', metrics = ['accuracy'])
#Fit the model with the training set and validation set
autoencoder.fit(x_train_Gray, x_train, validation_data = (x_test_Gray, x_test),epochs = 50, batch_size = 32, callbacks = callbacks)
x_decoded = autoencoder.predict(x_test_Gray)
autoencoder.save('colourization_model.h5')
```

```
Epoch 1/50
1563/1563 [==============================] - ETA: 0s - loss: 0.0151 - accuracy: 0.4832
Epoch 00001: val_loss improved from inf to 0.01070, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 14s 9ms/step - loss: 0.0151 - accuracy: 0.4832 - val_loss: 0.0107 - val_accuracy: 0.5101
Epoch 2/50
1562/1563 [============================>.] - ETA: 0s - loss: 0.0098 - accuracy: 0.5067
Epoch 00002: val_loss improved from 0.01070 to 0.00929, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 14s 9ms/step - loss: 0.0098 - accuracy: 0.5067 - val_loss: 0.0093 - val_accuracy: 0.5173
Epoch 3/50
1563/1563 [==============================] - ETA: 0s - loss: 0.0089 - accuracy: 0.5127
Epoch 00003: val_loss improved from 0.00929 to 0.00865, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 14s 9ms/step - loss: 0.0089 - accuracy: 0.5127 - val_loss: 0.0087 - val_accuracy: 0.5145
Epoch 4/50
1560/1563 [============================>.] - ETA: 0s - loss: 0.0085 - accuracy: 0.5185
Epoch 00004: val_loss improved from 0.00865 to 0.00845, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 14s 9ms/step - loss: 0.0085 - accuracy: 0.5185 - val_loss: 0.0084 - val_accuracy: 0.5302
Epoch 5/50
1559/1563 [============================>.] - ETA: 0s - loss: 0.0081 - accuracy: 0.5236
Epoch 00005: val_loss improved from 0.00845 to 0.00829, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 14s 9ms/step - loss: 0.0081 - accuracy: 0.5236 - val_loss: 0.0083 - val_accuracy: 0.4915
Epoch 6/50
1562/1563 [============================>.] - ETA: 0s - loss: 0.0078 - accuracy: 0.5274
```

```
Epoch 7/50
1561/1563 [============================>.] - ETA: 0s - loss: 0.0075 - accuracy: 0.5328
Epoch 00007: val_loss improved from 0.00800 to 0.00773, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 14s 9ms/step - loss: 0.0075 - accuracy: 0.5328 - val_loss: 0.0077 - val_accuracy: 0.5440
Epoch 8/50
1557/1563 [============================>.] - ETA: 0s - loss: 0.0073 - accuracy: 0.5376
Epoch 00008: val_loss improved from 0.00773 to 0.00760, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 14s 9ms/step - loss: 0.0073 - accuracy: 0.5376 - val_loss: 0.0076 - val_accuracy: 0.5311
Epoch 9/50
1562/1563 [============================>.] - ETA: 0s - loss: 0.0071 - accuracy: 0.5424
Epoch 00009: val_loss improved from 0.00760 to 0.00757, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0071 - accuracy: 0.5424 - val_loss: 0.0076 - val_accuracy: 0.5239
Epoch 10/50
1558/1563 [============================>.] - ETA: 0s - loss: 0.0069 - accuracy: 0.5462
Epoch 00010: val_loss improved from 0.00757 to 0.00746, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0069 - accuracy: 0.5461 - val_loss: 0.0075 - val_accuracy: 0.5169
Epoch 11/50
1559/1563 [============================>.] - ETA: 0s - loss: 0.0067 - accuracy: 0.5521
Epoch 00011: val_loss did not improve from 0.00746
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0067 - accuracy: 0.5520 - val_loss: 0.0076 - val_accuracy: 0.5063
Epoch 12/50
1563/1563 [==============================] - ETA: 0s - loss: 0.0065 - accuracy: 0.5571
Epoch 00012: val_loss did not improve from 0.00746
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0065 - accuracy: 0.5571 - val_loss: 0.0075 - val_accuracy: 0.5373
Epoch 13/50
1559/1563 [============================>.] - ETA: 0s - loss: 0.0063 - accuracy: 0.5614
Epoch 00013: val_loss did not improve from 0.00746
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0063 - accuracy: 0.5615 - val_loss: 0.0075 - val_accuracy: 0.5317
Epoch 14/50
1560/1563 [============================>.] - ETA: 0s - loss: 0.0061 - accuracy: 0.5663
Epoch 00014: val_loss did not improve from 0.00746
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0061 - accuracy: 0.5663 - val_loss: 0.0075 - val_accuracy: 0.5428
Epoch 15/50
1561/1563 [============================>.] - ETA: 0s - loss: 0.0058 - accuracy: 0.5722
Epoch 00015: ReduceLROnPlateau reducing learning rate to 0.00031622778103685084.
```

```
Epoch 00015: val_loss did not improve from 0.00746
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0058 - accuracy: 0.5721 - val_loss: 0.0077 - val_accuracy: 0.4824
Epoch 16/50
1562/1563 [============================>.] - ETA: 0s - loss: 0.0051 - accuracy: 0.5901
Epoch 00016: val_loss improved from 0.00746 to 0.00740, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0051 - accuracy: 0.5901 - val_loss: 0.0074 - val_accuracy: 0.5253
Epoch 17/50
1561/1563 [============================>.] - ETA: 0s - loss: 0.0048 - accuracy: 0.5956
Epoch 00017: val_loss improved from 0.00740 to 0.00734, saving model to /content/saved_models/colorized_ae_model.h5
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0048 - accuracy: 0.5956 - val_loss: 0.0073 - val_accuracy: 0.5181
Epoch 18/50
1561/1563 [============================>.] - ETA: 0s - loss: 0.0047 - accuracy: 0.5998
Epoch 00018: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0047 - accuracy: 0.5998 - val_loss: 0.0075 - val_accuracy: 0.5254
Epoch 19/50
1560/1563 [============================>.] - ETA: 0s - loss: 0.0046 - accuracy: 0.6031
Epoch 00019: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0046 - accuracy: 0.6032 - val_loss: 0.0074 - val_accuracy: 0.5192
Epoch 20/50
1557/1563 [============================>.] - ETA: 0s - loss: 0.0045 - accuracy: 0.6076
Epoch 00020: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0045 - accuracy: 0.6074 - val_loss: 0.0075 - val_accuracy: 0.5098
Epoch 21/50
1559/1563 [============================>.] - ETA: 0s - loss: 0.0044 - accuracy: 0.6111
Epoch 00021: val_loss did not improve from 0.00734
1563/1563 [==============================] - 14s 9ms/step - loss: 0.0044 - accuracy: 0.6110 - val_loss: 0.0076 - val_accuracy: 0.4966
Epoch 22/50
1560/1563 [============================>.] - ETA: 0s - loss: 0.0043 - accuracy: 0.6140
Epoch 00022: ReduceLROnPlateau reducing learning rate to 0.0001000000063960619 9.

Epoch 00022: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0043 - accuracy: 0.6140 - val_loss: 0.0076 - val_accuracy: 0.5017
Epoch 23/50
1557/1563 [============================>.] - ETA: 0s - loss: 0.0040 - accuracy: 0.6239
Epoch 00023: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0040 - accuracy: 0.6239 - val_loss: 0.0075 - val_accuracy: 0.5105
```

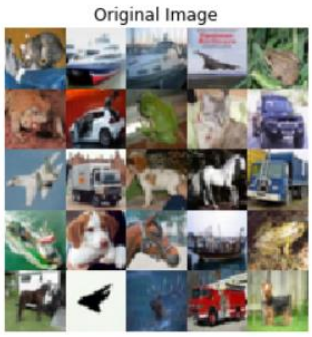
```
Epoch 24/50
1558/1563 [============================>.] - ETA: 0s - loss: 0.0039 - accuracy: 0.6259
Epoch 00024: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0039 - accuracy: 0.6259 - val_loss: 0.0076 - val_accuracy: 0.5191
Epoch 25/50
1561/1563 [============================>.] - ETA: 0s - loss: 0.0039 - accuracy: 0.6279
Epoch 00025: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0039 - accuracy: 0.6278 - val_loss: 0.0076 - val_accuracy: 0.5169
Epoch 26/50
1557/1563 [============================>.] - ETA: 0s - loss: 0.0039 - accuracy: 0.6292
Epoch 00026: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0039 - accuracy: 0.6293 - val_loss: 0.0076 - val_accuracy: 0.5124
Epoch 27/50
1557/1563 [============================>.] - ETA: 0s - loss: 0.0038 - accuracy: 0.6310
Epoch 00027: ReduceLROnPlateau reducing learning rate to 3.1622778103685084e-05.

Epoch 00027: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0038 - accuracy: 0.6310 - val_loss: 0.0076 - val_accuracy: 0.5087
Epoch 28/50
1561/1563 [============================>.] - ETA: 0s - loss: 0.0037 - accuracy: 0.6349
Epoch 00028: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0037 - accuracy: 0.6348 - val_loss: 0.0077 - val_accuracy: 0.5137
Epoch 29/50
1558/1563 [============================>.] - ETA: 0s - loss: 0.0037 - accuracy: 0.6358
Epoch 00029: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0037 - accuracy: 0.6357 - val_loss: 0.0077 - val_accuracy: 0.5057
Epoch 30/50
1562/1563 [============================>.] - ETA: 0s - loss: 0.0037 - accuracy: 0.6363
Epoch 00030: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0037 - accuracy: 0.6362 - val_loss: 0.0077 - val_accuracy: 0.5133
Epoch 31/50
1559/1563 [============================>.] - ETA: 0s - loss: 0.0037 - accuracy: 0.6368
Epoch 00031: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0037 - accuracy: 0.6367 - val_loss: 0.0077 - val_accuracy: 0.5075
Epoch 32/50
1559/1563 [============================>.] - ETA: 0s - loss: 0.0037 - accuracy: 0.6373
```

```
Epoch 33/50
1561/1563 [============================>.] - ETA: 0s - loss: 0.0037 - accuracy: 0.6386
Epoch 00033: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0037 - accuracy: 0.6387 - val_loss: 0.0077 - val_accuracy: 0.5108
Epoch 34/50
1557/1563 [============================>.] - ETA: 0s - loss: 0.0037 - accuracy: 0.6390
Epoch 00034: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0037 - accuracy: 0.6389 - val_loss: 0.0077 - val_accuracy: 0.5099
Epoch 35/50
1560/1563 [============================>.] - ETA: 0s - loss: 0.0037 - accuracy: 0.6391
Epoch 00035: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0037 - accuracy: 0.6391 - val_loss: 0.0077 - val_accuracy: 0.5075
Epoch 36/50
1562/1563 [============================>.] - ETA: 0s - loss: 0.0037 - accuracy: 0.6392
Epoch 00036: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0037 - accuracy: 0.6392 - val_loss: 0.0077 - val_accuracy: 0.5089
Epoch 37/50
1562/1563 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 0.6395
Epoch 00037: ReduceLROnPlateau reducing learning rate to 3.1622778678900043e-06.

Epoch 00037: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0036 - accuracy: 0.6396 - val_loss: 0.0077 - val_accuracy: 0.5085
Epoch 38/50
1561/1563 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 0.6400
Epoch 00038: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0036 - accuracy: 0.6400 - val_loss: 0.0077 - val_accuracy: 0.5087
Epoch 39/50
1557/1563 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 0.6401
Epoch 00039: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0036 - accuracy: 0.6401 - val_loss: 0.0077 - val_accuracy: 0.5086
Epoch 40/50
1563/1563 [==============================] - ETA: 0s - loss: 0.0036 - accuracy: 0.6401
Epoch 00040: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0036 - accuracy: 0.6401 - val_loss: 0.0077 - val_accuracy: 0.5076
Epoch 41/50
1559/1563 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 0.6401
```

```
Epoch 00042: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0036 - accuracy: 0.6402 - val_loss: 0.0077 - val_accuracy: 0.5091
Epoch 43/50
1557/1563 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 0.6402
Epoch 00043: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0036 - accuracy: 0.6402 - val_loss: 0.0077 - val_accuracy: 0.5093
Epoch 44/50
1558/1563 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 0.6404
Epoch 00044: val_loss did not improve from 0.00734
1563/1563 [==============================] - 14s 9ms/step - loss: 0.0036 - accuracy: 0.6404 - val_loss: 0.0077 - val_accuracy: 0.5093
Epoch 45/50
1561/1563 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 0.6404
Epoch 00045: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0036 - accuracy: 0.6404 - val_loss: 0.0077 - val_accuracy: 0.5094
Epoch 46/50
1562/1563 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 0.6403
Epoch 00046: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0036 - accuracy: 0.6403 - val_loss: 0.0077 - val_accuracy: 0.5093
Epoch 47/50
1563/1563 [==============================] - ETA: 0s - loss: 0.0036 - accuracy: 0.6405
Epoch 00047: ReduceLROnPlateau reducing learning rate to 5e-07.

Epoch 00047: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0036 - accuracy: 0.6405 - val_loss: 0.0077 - val_accuracy: 0.5088
Epoch 48/50
1562/1563 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 0.6403
Epoch 00048: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0036 - accuracy: 0.6404 - val_loss: 0.0077 - val_accuracy: 0.5094
Epoch 49/50
1559/1563 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 0.6405
Epoch 00049: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0036 - accuracy: 0.6405 - val_loss: 0.0077 - val_accuracy: 0.5094
Epoch 50/50
1563/1563 [==============================] - ETA: 0s - loss: 0.0036 - accuracy: 0.6405
Epoch 00050: val_loss did not improve from 0.00734
1563/1563 [==============================] - 13s 9ms/step - loss: 0.0036 - accuracy: 0.6405 - val_loss: 0.0077 - val_accuracy: 0.5093
```

Loading the model and checking out the results

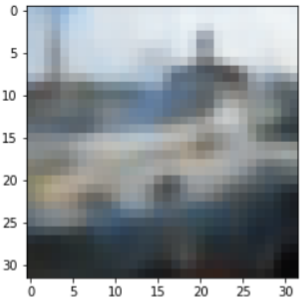
```
[ ] from google.colab import files
    #files.download('colourization_model.h5')
    #Displaying Results
    imgs = x_test[:25]
    imgs = imgs.reshape((5, 5, img_dim, img_dim, channels))
    imgs = np.vstack([np.hstack(i) for i in imgs])
    plt.figure()
    plt.axis('off')
    plt.title('Original Image')
    plt.imshow(imgs, interpolation='none')
    #plt.savefig('%s/colorized.png' % imgs_dir)
    plt.show()
    imgs = x_decoded[:25]
    imgs = imgs.reshape((5, 5, img_dim, img_dim, channels))
    imgs = np.vstack([np.hstack(i) for i in imgs])
    plt.figure()
    plt.axis('off')
    plt.title('Colorized test images (Predicted)')
    plt.imshow(imgs, interpolation='none')
    #plt.savefig('%s/colorized.png' % imgs_dir)
    plt.show()
```




```
[ ]  plt.imshow(x_decoded[2])
```

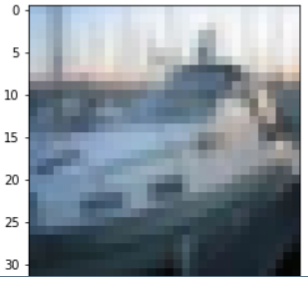
```
[ ]  plt.imshow(x_decoded[2])
```

```
<matplotlib.image.AxesImage at 0x7f8338ebb668>
```

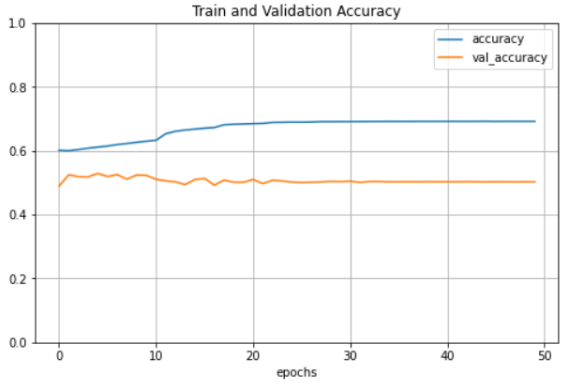



```
[ ]  plt.imshow(x_test[2])
```

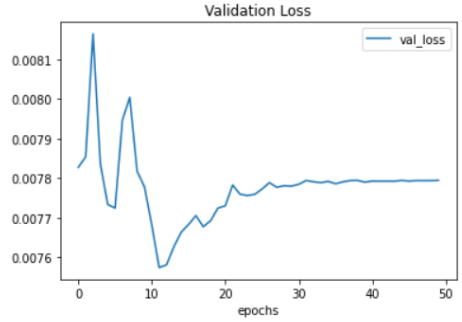
```
<matplotlib.image.AxesImage at 0x7f8338d07da0>
```



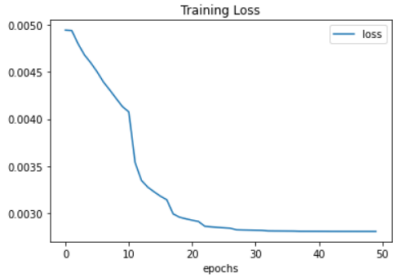
```
[ ] df = pd.DataFrame(history.history)
    df[['accuracy','val_accuracy']].plot(figsize=(8, 5))
    plt.grid(True)
    plt.gca().set_ylim(0, 1) # set the vertical range to [0-1]
    plt.xlabel("epochs")
    plt.title('Train and Validation Accuracy')
    plt.show()
```




```
[ ] df[['val_loss']].plot()
    plt.xlabel("epochs")
    plt.title('Validation Loss')
    plt.show()
```




```
df[['loss']].plot()
plt.xlabel("epochs")
plt.title('Training Loss')
plt.show()
```




```
[ ]
```

CONCLUSION

Through the amalgamation of decoder and encoder models, we were successful in exploring the space of convolutional neural networks and auto-encoders. Our model successfully converted a single-channel black and white image to its corresponding RGB layered colorized form automatically without any modifications to its inherent physical properties. We were able to understand the importance of regularization techniques and also why they were more harmful towards our model rather that providing better results. This was due to their normalizing and mean-wrapped behaviour which resulted in muddy and desaturated images. Through a vanilla autoencoder model, we were able to produce vivid images with low loss (0.3-0.4) and high accuracy (50-52%) that most closely resembled the original images before pre-processing.

References

[1] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycleconsistent adversarial networks." arXiv preprint arXiv:1703.10593 (2017).

[2] Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." arXiv preprint arXiv:1611.07004 (2016).

[3] Reed, Scott, et al. "Generative adversarial text to image synthesis." 33rd International Conference on Machine Learning. Vol. 3. 2016.

[4] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).

[5] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." (2014).

[6] Hariharan, Bharath, et al. "Hypercolumns for object segmentation and fine-grained localization." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015

[7] Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification." ACM Transactions on Graphics (TOG) 35.4 (2016): 110.

[8] Larsson, Gustav, Michael Maire, and Gregory Shakhnarovich. "Learning representations for automatic colorization." European Conference on Computer Vision. Springer International Publishing, 2016.

[9] Charpiat, Guillaume, Matthias Hofmann, and Bernhard Schölkopf. "Automatic image colorization via multimodal predictions." Computer Vision– ECCV 2008 (2008): 126-139.

[10] Deshpande, Aditya, Jason Rock, and David Forsyth. "Learning large-scaleautomatic image colorization." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[11] Welsh, Tomihisa, Michael Ashikhmin, and Klaus Mueller. "Transferring color to greyscale images." ACM Transactions on Graphics (TOG). Vol. 21. No.3. ACM, 2002.

[12] Qu, Yingge, Tien-Tsin Wong, and Pheng-Ann Heng. "Manga colorization."ACM Transactions on Graphics (TOG). Vol. 25. No. 3. ACM, 2006.

[13] Huang, Yi-Chin, et al. "An adaptive edge detection based colorization algorithm and its applications." Proceedings of the 13th annual ACM international conference on Multimedia. ACM, 2005.

[14] Levin, Anat, Dani Lischinski, and Yair Weiss. "Colorization using optimization." ACM Transactions on Graphics (ToG). Vol. 23. No. 3. ACM, 2004.

[15] Li, Jiwei, et al. "Adversarial learning for neural dialogue generation." arXiv preprint arXiv:1701.06547 (2017).

[16] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

[17] Dahl, Ryan. "Automatic colorization." (2016).

[18] Cheng, Zezhou, Qingxiong Yang, and Bin Sheng. "Deep colorization." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[19] Liang, Xiangguo, et al. "Deep patch-wise colorization model for grayscale images." SIGGRAPH ASIA 2016 Technical Briefs. ACM, 2016.

[20] Zhang, Richard, Phillip Isola, and Alexei A. Efros. "Colorful image colorization." European Conference on Computer Vision. Springer International Publishing, 2016.

[21] Deep Colorization Zezhou Cheng, Student Member, IEEE, Qingxiong Yang, Member, IEEE, Bin Sheng, Member, IEEE, arXiv:1605.00075v1

[22] Interactive Anime Sketch Colorization with Style Consistency via aDeep Residual Neural Network Ru-Ting Ye,Wei-Li Wang,Ju-Chin Chen Kawuu W. Lin [23] Learning-Based Colorization of Grayscale Aerial Images Using Random Forest Regression Dae Kyo Seo, Yong Hyun Kim , Yang Dam Eo and Wan Yong Park

[23] Automatic Image Colorization Via Multimodal Predictions Guillaume Charpiat, Matthias Hofmann, and Bernhard Sch¨olkopf

[24] Semi-Auto Sketch Colorization Based on Conditional Generative Adversarial Networks - Zijuan Cheng Fang Meng Jingbo Mao

[25] Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

[26] Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1

[27] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[28] M. M. Hadhoud, N. A. Semary and A. M. Abbas, "Fully automated black and white movies colorization system

[29] Y. Xiao, A. Jiang, C. Liu and M. Wang, "Single Image Colorization Via Modified Cyclegan," 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 2019

[30] H. Brendan Mcmahan and Matthew Streeter. Delay-Tolerant Algorithms for Asynchronous Distributed Online Learning. Advances in Neural Information Processing Systems (Proceedings of NIPS), pages 1–9, 2014.

[31] 1.Rainer Böhme, Matthias Kirchner, H. T. Sencar and N. Memon, "Counter-forensics: Attacking image forensics" in Digital Image Forensics, Springer Berlin/Heidelberg, 2012.

[32] Mauro Barni and Fernando Pérez-González, "Coping with the enemy: advances in adversary-aware signal processing", *ICASSP 2013 IEEE International Conference on Acoustics Speech and Signal Processing*, pp. 8682-8686, 26-31 May 2013.

[33] Matthew C. Stamm and K.J. Ray Liu, "Forensic detection of image manipulation using statistical intrinsic fingerprints", *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 3, pp. 492-506, 2010.

[34] Gang Cao, Yao Zhao, Rongrong Ni and Huawei Tian, "Anti-forensics of contrast enhancement in digital images", *Proceedings of the 12th ACM Workshop on Multimedia and Security*, pp. 25-34, 2010

[35] Alessia De, Marco Rosa, Matteo Fontani, Alessandro Massai, Piva and Mauro Barni, "Second-order statistics analysis to cope with contrast enhancement counter-forensics", *IEEE Signal Processing Letters*, vol. 22, no. 8, pp. 1132-1136, 2015.

[36] Mauro Barni, Ehsan Nowroozi and Benedetta Tondi, "Higher-order adversary-aware double JPEG-detection via selected training on attacked samples", *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 281-285, Aug 2017.

[37] Belhassen Bayar and Matthew C. Stamm, "A deep learning appraoch to universal image manipulation detection using a new convolutional layer", *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*.

[38] Karen Simonyan and Andrew Zisserman, *Very deep convolutional networks for large-scale image recognition*, pp. 5-10, 2014.

[39] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter and Giulia Boato, "Raise: A raw images dataset for digital image forensics", *Proceedings of the 6th ACM Multimedia Systems Conference*, pp. 219-224, 2015.

[40] Karen Simonyan and Andrew Zisserman, *Very deep convolutional networks for large-scale image recognition*, pp. 5-10, 2014.