# Sillystill: Recreating the Look of Cinestill-800T using Deep Learning

Mika Senghaas  Annamira O'Toole  Pierre Lardet

Supervised by Dr. Raphaël Achddou in CS413 at EPFL

**Abstract.** The use of deep learning in stylistic effect generation has seen increasing use over recent years. In this work, we use simple convolutional neural networks to model Cinestill800T film given a digital input. We test the effect of different loss functions, as well as the addition of an input noise channel and the use of random scales of patches during training. We find that a combination of MSE/VGG gives the best colour production, and that some grain can be produced, but it is not of a high quality, and no halation is produced. We contribute our dataset of aligned paired images taken with a film and digital camera for further work.

## 1 Introduction

Film photography is popular among artists and photographers for capturing scenes in a unique way. They use optical flaws, such as halation and chromatic aberrations, for aesthetic purposes. Such visual effects are the result of complex physical processes that are not present in modern digital photography. Hence, to recreate the visual effects of film in digital images, computational methods are required. This motivates our work, where we investigate the feasibility of using a pure deep learning approach to recreate the visual effects of Cinestill-800T film, an iconic film stock known for its grain textures, color hues and red halation.

Most available solutions involve directly modelling the physical processes that lead to the visual effects [22]. However, the complex interplay between the camera, film stock, and chemicals during the development process, means that simplifying assumptions have to be made. In addition, such models are often computationally expensive and time consuming, especially when applied to high-resolution images.

Ideally, a digital image could be processed into its film equivalent on the order of seconds with high fidelity. Deep learning models have shown impressive results in learning complex mappings, out-performing handcrafted models in many tasks. One can hope that a deep learning model could learn higher quality style translations than existing methods, while reducing computational complexity. However, the problem of recreating film effect has not been directly addressed by deep learning literature. Related computational photography problems include camera raw-to-raw mappings, quality enhancement, grayscale colorisation, and style transfer. As none of these involve direct applicatino to the problem of film

effect generation, we examine the commonly used techniques and how they could be adapted to our task.

All of this motivates our work, where we investigate the feasibility of using deep learning to recreate the visual effects of Cinestill-800T film in digital images. We propose training a U-Net-like model translation network on a new dataset of paired digital and film images. While we acknowledge that using direct statistical models for film grain, halation, and color hue may be effective as a standalone, or in combination with deep learning, we constrain this work to only explore pure deep learning methods.

Our contributions can be summarised as:

- We propose a pure deep learning approach to recreate the visual effects of Cinestill-800T film in digital images.
- We present a self-collected dataset of paired digital and film images, which we publicly release to the research community.
- We investigate and analyse various techniques that help to improve the creation of visual effects, such as adding noise to the input, training on random-resized crops, or varying the loss function.

## 2 Literature Review

Many methods have presented as viable solutions for stylistic effect generation using image to image translation using Deep Learning. Among these, there is a variety of kinds of model architectures and effects produced.

**Image Translation.** GANs have seen successful application in generic image translation, such as in Conditional Image-Image Translation [10] in which a loss is learned by a discriminator network which is conditioned on the input. This can be generalised to unpaired image data using Cyclic GANs [32] where two translation networks are used in either direction, and cyclic consistency is enforced using the loss function.

**Super Resolution.** The task of generating a higher resolution image from a lower resolution image has been tackled by deep learning approaches with a variety of pixel-wise and perceptual losses [5, 11, 15]. GANs have produced impressive results [16, 28], especially in combination with perceptual losses [20, 27]. Methods for greater stability during training have been developed [13] and for greater control over which style is produced [14].

A contextual bilateral loss was introduced by [31] based on RGB and VGG feature patches that builds on the contextual loss proposed by [19] by adding a distance weighting. The authors claim that this loss is robust to misalignment in paired datasets. We implemented this loss but found it did not improve our results and do not include it in our final models.

**Image Colorisation.** Converting a grayscale image to a color has been attempted by [4] using chromaticity maps, and [17] addresses the problem of illuminants by searching the web for reference images. [3] leverages deep learning with separate networks for different kinds of images. This suggests a natural extension of our work would be to condition the digital-film mapping on features such as a luminance histogram.

**Raw to Raw Mapping.** [1] maps the raw output of one camera to another using paired auto-encoders. Each auto-encoder is trained on raw outputs from a different camera, and the latent space of the two auto-encoders is mapped using a paired dataset. We implemented this architecture for a digital to film mapping, but it did not produce good initial results so we did not pursue it further.

**Style Transfer.** Style transfer, which maps the style of one image to the content of another, has seen success using CNNs which are trained to minimize a content and style loss. This has been applied to paintings [6, 7] and to photorealistic images [18]. These methods require an expensive optimization for every image. Instead, [11] uses perceptual losses based on VGG features to train a feed forward model and [8] extends this to arbitrary styles using adaptive instance normalization. [9] attempts to convert the style of phone images to DSLR quality using a composite loss including functions for colour, texture and content. We follow a similar approach in our work, using perceptual features produced by VGG-19 [26] as part of our loss to train a feed forward CNN for style transfer.

## 3 Implementation

To recreate the look of Cinestill-800T we aim to train a deep image-to-image translation network on a paired digital-film dataset. This requires a dataset of digital images and their corresponding film images, a suitable model architecture, and loss functions capable of capturing the appearance of film.

### 3.1 Data Collection and Preprocessing

While unpaired digital or film datasets are abundant, our method requires paired digital-film images. This means that the exact same scene has to be captured with both a digital and a film camera to allow the model to learn the differences in appearance between the two media. Images should be aligned, as our model should only transfer style and not learn move contents of the image around, and diverse in order to capture Cinestill-800T's baheviour in a variety of scenes and lighting conditions. As no such dataset was available, we collected our own.

**Data Collection.** To ensure aligned images, we take inspiration from the setup proposed in [9] and capture images with two cameras, a Sony Alpha 7 digital camera and a Nikon F3 film camera. Photos were taken by keeping the same tripod setup for both cameras for each image. Camera settings like aperture, ISO and shutter speed were kept constant across both cameras as best as possible (Table 1). We only captured static scenes to limit changes in the scene

between shots. To create a diverse dataset, images were taken both indoors and outdoors, in different weather conditions, with different light and containing different subjects. Specific emphasis was put on ensuring that all special visual effects (halation, grain and colours) were well-represented in the dataset. The final raw dataset consists of 41 image pairs. Some examples are presented in Figure 1.

**Table 1:** Camera Setup.

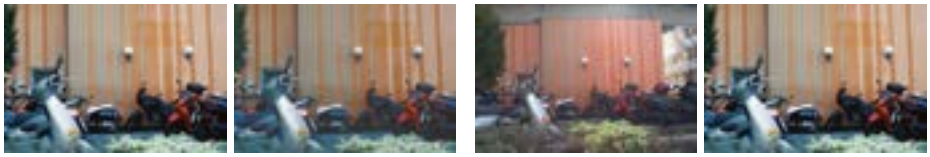
| Camera | Resolution | Lens | Aperture | ISO | Shutter |
|---|---|---|---|---|---|
| Sony Alpha 7 (Digital) | 24 MP | 40mm | F8 | 100 | Auto |
| Nikon F3 (Film) | 12 MP | 55mm | F8 | 100 | Auto |

**Fig. 1: Raw Paired Image Dataset.** Examples of raw image pairs from the dataset. A column shows a single scene captured with a digital camera (top) and a film camera (bottom). The images show a wide range of scenes and visual effects.

**Data Preprocessing** Inevitably, raw image pairs are not perfectly aligned. To address this issue, we first spatially align the images using keypoint alignment. We follow a standard processing pipeline using openly available implementations of common feature detection, matching and perspective transformation algorithms from the OpenCV library [2]. For all image pairs we first detect keypoints and extract descriptors using ORB [25], match them using FLANN [21], and estimate a homography transformation matrix using the RANSAC algorithm.

To ensure that our model does not learn systematic brightness adjustments, we also align the luminance of each image pair. We align the luminance of the film image to the digital image as our model receives digital as input at inference time. The luminance alignment is performed using histogram matching on the luminance channel in CIELAB space before converting back to RGB.

An example of the results of these two steps can be seen in Figure 2. After the preprocessing steps we are left with 38 processed image pairs. We publicly release both the raw and processed dataset to facilitate further work.



**(a)** Film: Raw (left) vs. Processed (right). **(b)** Digital: Raw (left) vs. Processed (right).

**Fig. 2: Dataset Preprocessing.** Example of a raw and processed image pair. We can see the luminance alignment in the film image (a) and the spatial alignment in the digital image (b).

### 3.2 Model

Our image style translation network is based on the U-Net architecture [24]. U-Net is a type of convolutional neural network (CNN) originally introduced for image segmentation tasks in 2015. CNNs have been successfully adapted for various image-to-image translation tasks and similar models are used as the building block of many modern model families, such as auto-encoders [1] and GANs [12]. The network features a U-shaped architecture, composed of an encoder and a decoder. The encoder resembles a conventional convolutional network, involving repeated convolutions followed by a non-linear activation functions and pooling operations. The decoder integrates both feature and spatial information using up-convolutions and high-resolution features via skip connections.

We selected UNet as it is a relatively simple model family, allowing us to focus on the specifics of our problem rather than the intricacies of the model. Despite its simplicity, similar models such as CNNs with residual connections have been shown to be effective for image-to-image translation, such as in [9]. As it is fully convolutional, it can be applied to images of arbitrary size. [1]

For our experiments we follow related work [1, 9, 24] to set sensible hyperparameters for our model. Our image style translation network is fully convolutional, with a three-layer encoder and decoder. The encoder consists of three convolutional blocks with 64, 128, and 256 filters, respectively. Each block consists of two convolutional layers with 3×3 kernels and zero padding, interleaved with ReLu activation functions. In between each of the convolutional blocks, we downsample the spatial dimension of the feature maps by a factor of two using a 2×2 max-pooling operation. The decoder performs the reverse operation by upsampling the spatial dimension. It decreases the number of filters from 256 to 128,

[1] So long as the input dimensions are divisible by $2^L$, where $L$ is the number of layers in the encoder

64, and finally to 3. Upsampling is performed via a 2×2 transposed convolution. After upsampling, we concatenate the feature maps from the corresponding layer in the encoder before applying a convolutional block. After the final layer, the model outputs a three-channel image, corresponding to the RGB channels of the transformed image. Thus, the overall translation network $F_\theta$ is an image-to-image model, that takes an image $I$ as input and outputs a transformed image $F_\theta(I)$ of the same dimensions.

### 3.3 Loss Functions

We propose a list of loss functions that can be combined, each targeting different effects.

**MSE/MAE.** Both the mean squared error (MSE, Eq. 1) and mean absolute error (MAE, Eq. 2) are common pixel-to-pixel loss functions, that are popular in image reconstruction tasks. They measure the average squared or absolute difference between the predicted and ground truth images pixels, respectively. For two images, $X, Y \in \mathbb{R}^{3\times H\times W}$, the MSE and MAE losses are defined as:

$$\mathcal{L}_{\text{MSE}}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i, Y_i)^2 \quad (1) \qquad \mathcal{L}_{\text{MAE}}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}|X_i - Y_i| \quad (2)$$

where $i$ indexes the pixels of the images. We hypothesise that such loss functions are useful in preserving the overall structure and color of the image, but are not sufficient to capture higher-level visual effects such as grain and halation.

**VGG Loss.** VGG loss computes the MSE (Eq. 1) between the feature representations of the predicted and ground truth images when passed through a pre-trained VGG-19 [26] convolutional network. Letting $\varphi_k(I)$ denote the feature maps of the $k$-th layer of the VGG network, and the sets $K$ and $\lambda$ denote the layers of interest [2] and their weights in the loss function respectively, the VGG loss is defined as:

$$\mathcal{L}_{\text{VGG}}(X,Y) = \sum_{k\in K}\lambda_k\mathcal{L}_{\text{MSE}}(\varphi_k(X), \varphi_k(Y)). \quad (3)$$

Intuitively, the VGG loss encourages the model to generate images that are similar to the ground truth images in terms of their high-level semantics such as edges, textures and patterns.

**Color Loss.** Inspired by [9], we consider an alternative to the MSE loss, which we refer to as the color loss. The color loss computes the MSE between two images, $X_b$ an $Y_b$, where $X_b$ and $Y_b$ are blurred versions of the predicted and

[2] We use conv1_2, conv2_2, conv3_2 with weights 0.4, 0.4 and 0.2 respectively.

ground truth images respectively. Blurring is done using a Gaussian filter with a kernel size of 7 and $\sigma = 3$. We can write:

$$\mathcal{L}_{\text{Color}}(X, Y) = \mathcal{L}_{\text{MSE}}(X_b, Y_b). \tag{4}$$

As the authors of the original paper argue, the main idea behind the loss is to evaluate the brightness, contrast and major colours of the image while ignoring fine-grained texture and content comparison. Crucially, the color loss, unlike MSE or MAE, is more robust to small pixel shifts which are present in our training data.

**Relative Total Variational Loss (TV-Rel).** Traditionally, the total variational loss is used to encourage image smoothness and reduce noise by minimising the sum of the absolute differences between shifted pixel values. In our case, we aim to create grain, which can be seen as a form of noise. To this end, we propose the relative total variational loss defined as the absolute difference between the total variation of the predicted and ground truth images.

$$\mathcal{L}_{\text{TV-Rel}} = |TV(X) - TV(Y)| \tag{5}$$

This loss encourages the model to generate images with a similar amount of noise to the ground truth film images.

**Combination of Losses.** We combine the individual loss functions to form the final loss function using a weighted sum. For a set of loss functions $\mathcal{L}$ and corresponding weights $\beta$ [3], the final loss function is defined as

$$\mathcal{L}(X, Y) = \sum_i \beta_i \mathcal{L}_i(X, Y). \tag{6}$$

Some combinations of loss functions are more effective than others, and we will experiment with different combinations, as outlined in Section 4.

### 3.4 Experiments

The training procedure is illustrated in Figure 3. We train our model on a dataset of paired digital-film images. As our image dataset is relatively small, we use a patched training approach, where we randomly sample patches of size $256 \times 256$ from the images before feeding them into the model. We train our model on 400 patches per image in the dataset using the Adam optimizer with a learning rate of $1e - 3$ and a batch size of 1. Our experiments are divided into two parts: single-image experiments and full dataset experiments.

For our single-image experiments, we handpicked an image from our dataset that contains a white wall with a light source. It was chosen for the clear blue tint

[3] We simply use equal weights in all experiments with $\beta_i = 1$ for all $i$

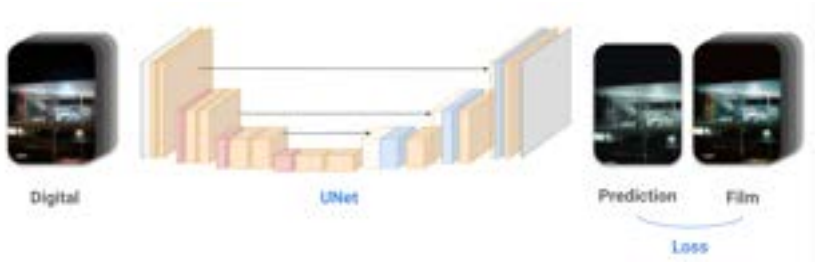



**Fig. 3: Training Overview.** We train our model on paired digital-film images.

of the film effect and the clearly perceptible grain on the uniform background. For these experiments, we train and evaluate the model on the same image. We first investigate how each loss produces the desired colour effect to gain a better understanding of the interplay between the loss functions and what the model learns.

For the most promising candidates from the single-image experiments, we then move on to training on the full dataset. The full dataset is split into a training, validation and test set using a 70-20-10 split ratio, but final evaluation is done on the full dataset.

We also investigate whether adding noise as a 4th input channel to the model improves the its ability to learn the desired visual effects for both single image and full dataset experiments. This is done by concatenating a random uniform noise [4] channel to the input images. Furthermore, we observed that patched training is very stochastic as each patch only captures a very small portion of the original image and can be very different from the rest of the image. To contain more information in each patch we also experiment applying a resized crop where we first crop the image to a larger patch and then resize to the desired input size. This way we can ensure that each patch contains more information from the original image.

### 3.5 Evaluation

To evaluate the generated film images, we conduct both qualitative and quantitative assessments. For quantitative evaluation, we use four image quality and image similarity metrics. We include a reference baseline comparison which is calculated using the original digital image as the prediction.

We use two standard algorithmic metrics, **Peak Signal-to-Noise Ratio (PSNR)**, which measures the ratio of the maximum possible signal power to the power of corrupting noise, and **Structural Similarity Index (SSIM)**, which compares structural, luminance, and contrast elements between an original and a

[4] We tried using other forms of noise such as Guassian noise but found that this made little difference.

reconstructed image in a range from -1 to 1. We also use two learned distance metrics, **Learned Perceptual Image Patch Similarity (LPIPS)** [30] and **PieApp** [23], where lower indicates higher similarity, which are trained on human-annotated data to align with human perception of image quality.

## 4 Results

A comprehensive list of all results and experiments can be found in the Appendix Section 7.2.

### 4.1 Single Image Results

Table 2 shows each loss configuration with and without an added noise channel and without resizing. We find that the best performing loss configuration is a combined loss of Color/VGG/TV-Rel, which produces the best SSIM score without noise, and the best PieAPP score in either setting. Notably, the combined losses all perform well, as they account for both the colour and grain effects. The colour-focused losses (MAE, MSE and Colour) all perform well alone, whereas all feature-based losses (VGG and TV-Rel) almost always perform below the baseline. Interestingly, LPIPS scores are inconsistent with the other metrics with the best performing configuration as the baseline and TV-Rel alone with and without noise respectively. This is contrary to what we would be expect as can be seen in a sample of results in Figure 4.

**(a)** Film
Ground truth

**(b)** MSE/VGG
no noise, resize
LPIPS: 0.35
SSIM: **0.71**

**(c)** MSE
no noise, no resize
LPIPS: 0.47
SSIM: 0.67

**(d)** TV-Rel
noise, no resize
LPIPS: **0.27**
SSIM: 0.27

**(e)** Baseline

LPIPS: 0.36
SSIM: 0.64

**Fig. 4: Single Image Select Samples.** Outputs from select loss functions and configurations of noise and resizing. We see that the best performing model is MSE/VGG with resizing, which produces the best colour effect. We also see that LPIPS scores are inconsistent with SSIM and with perception as we can intuitively see that MSE-VGG and MSE are better predicitions than the baseline and TV-Rel.

The addition of noise has mixed effects. The perceptual metrics significantly reward the addition of noise in almost all cases. SSIM and PSNR react differently depending on the loss: pixel-wise MSE/MAE scores increase slightly, whereas Color loss scores decrease. This can be interpreted as Color loss allowing more noise through the model which is penalised by metrics sensitive to incorrect random noise such as PSNR, where MAE/MSE will even out this noise to be

**Table 2: Single Image Losses.**
Results for each loss configuration with and without noise and no resizing.

| Loss | Without Noise | | | | With Noise | | | |
|---|---|---|---|---|---|---|---|---|
| | SSIM | PSNR | LPIPS | PieAPP | SSIM | PSNR | LPIPS | PieAPP |
| Color/VGG/TV-Rel | **0.68** | 19.67 | 0.45 | **1.42** | 0.67 | 18.01 | 0.37 | **1.14** |
| Color/VGG | **0.68** | **20.07** | 0.45 | 1.61 | **0.68** | 17.94 | 0.36 | 1.58 |
| MSE/VGG | 0.66 | 18.26 | 0.48 | 2.53 | 0.67 | **19.79** | 0.34 | 2.05 |
| MAE | 0.65 | 18.61 | 0.50 | 2.12 | **0.68** | 19.04 | 0.33 | 1.96 |
| Color | 0.65 | 18.45 | 0.50 | 2.99 | 0.63 | 15.66 | 0.43 | 2.50 |
| MSE | 0.65 | 18.29 | 0.50 | 2.42 | 0.67 | 19.26 | 0.32 | 1.89 |
| Baseline | 0.64 | 16.97 | **0.36** | 3.79 | 0.64 | 16.97 | 0.36 | 3.79 |
| TV-Rel | 0.17 | 9.84 | 0.60 | 4.84 | 0.27 | 11.11 | **0.27** | 2.20 |
| VGG | 0.16 | 9.10 | 0.62 | 4.50 | 0.06 | 8.71 | 0.64 | 5.11 |

closer to the ground truth, on average. All of this implies that the perceptual metrics better capture the output of noise by the model. Comparisons of outputs from the models with and without noise channels can be seen in Figure 5.

We also experiment with resizing cropped patches during training. This is done by taking a crop at a random scale from the image and resizing it to the patch size, $256 \times 256$. The results are in Table 3. We show results for only MSE and MSE/VGG as a representative example of how both pixel-wise and combined losses perform. We also experimented with a combination of noise and resizing but found that both performed better alone. We find that we reach peak SSIM and PSNR scores when resizing the cropped patches, and perceptual metrics also improve across the board.

We hypothesise that while resizing loses high-resolution information on grain and noise patterns, the model sees a greater variety of areas of the image in a single patch and so may capture the color transformation better which is rewarded by every metric. With the MSE/VGG loss, this colour transformation is a nearly perfect overfit as the outputs in 6 show. Furthermore, some grain is still produced when we feed noise into the model despite the resizing operation potentially removing grain, as can be seen in Figure 7.

**Table 3: Single Image Resizing.**
Effect of resizing the cropped patches on select losses (no noise).

| Loss | Resize | SSIM | PSNR | LPIPS | PieAPP |
|---|---|---|---|---|---|
| MSE/VGG | Yes | **0.71** | **23.19** | **0.35** | **1.62** |
| MSE/VGG | No | 0.66 | 18.26 | 0.48 | 2.53 |
| MSE | Yes | **0.67** | **20.03** | **0.47** | **1.77** |
| MSE | No | 0.65 | 18.29 | 0.50 | 2.42 |

**(a)** Color/VGG/TV-Rel **(b)** MSE/VGG **(c)** MSE **(d)** Film (Above) Digital (Below)

**Fig. 5: Single Image Noise Comparison.** Outputs from select models without noise (above) and with noise (below), no resizing. We see that when we feed noise into the model, the model learns to produce some variation, especially when the loss contains a feature-based metric like VGG. However, the grain is far from the desired effect.

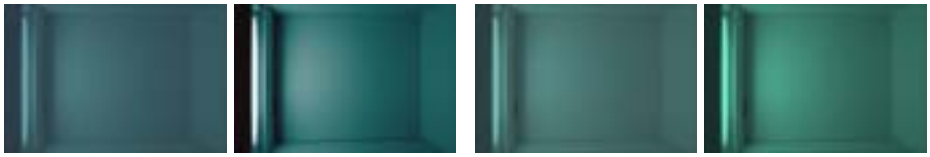
### 4.2 Full Dataset Results

We select the best performing losses and settings from the single image experiments and evaluate them on the full dataset, with results in Table 4.

**Table 4: Full Dataset Results.**
Comparison of best performing losses and configurations on the full dataset.

| Loss | Noise | Resize | SSIM | PSNR | LPIPS | PieAPP |
|---|---|---|---|---|---|---|
| Baseline | - | - | **0.64** | 21.68 | **0.26** | 1.96 |
| MSE/VGG | Yes | Yes | **0.64** | **22.87** | 0.27 | **1.90** |
| MSE | No | Yes | 0.63 | 22.20 | 0.35 | 2.31 |
| MAE | No | No | 0.59 | 19.68 | 0.48 | 3.38 |
| Color/VGG/TV-Rel | Yes | No | 0.59 | 21.42 | 0.41 | 3.07 |
| MSE/VGG/TV-Rel | Yes | No | 0.58 | 20.43 | 0.46 | 3.54 |

We selected models in terms of high scoring metrics and qualitative evaluation of color and grain production. This leaves us with some models that should produce the best colour: MSE/VGG and MSE with resizing, and other models that should produce the best grain: Color/VGG/TV-Rel, MSE/VGG/TV-Rel with a noise input channel. We find that the best performing model is MSE/VGG with resizing which produces the best metrics across the board. A notable difference compared to the single image results is that our methods now almost



**(a)** MSE/VGG, no noise
no resize (left) and with resizing (right).

**(b)** MSE, no noise
no resize (left) and with resizing (right).

**Fig. 6: Single Image Resizing.** Outputs from MSE and MSE/VGG with and without resizing, and then with resizing and with and without noise. We see that resizing improves the colour effect, but the grain effect is not as strong as in the single image experiments.

**(a)** MSE/VGG patch with resizing
no noise (left) and with noise (right).

**(b)** MSE patch with resizing
no noise (left) and with noise (right).

**Fig. 7: Single Image Resizing with Noise.** Outputs from MSE and MSE/VGG with and without noise and resizing. We see that similar grain is still produced even through resizing.

never outperform the baseline despite training on the same number of patches per image. This is likely due to the increased complexity of the full dataset, which contains a wider variety of film effects and lighting conditions. The only exception is MSE/VGG which is on par with the baseline. A sample of results can be seen in Figure 8.

We continue experimenting with the presence of resizing and noise using MSE and MSE/VGG, with results found in Table 5. Results indicate that once again resizing improves results, but the addition of noise has a less clear effect. The overall best performing model is MSE/VGG with resizing. Noise has little effect. However, the qualitative results tell a different story. In Figure 9 we see that the model with noise produces a more visually pleasing result, with a more uniform grain effect. This is not captured by the metrics, which are more sensitive to the colour effect.

Comparing results with the related task of enhancement of mobile phone images in [9] indicates that our models have a much lower SSIM, but comparable PSNR. They achieve peak SSIM of 0.94 where we observe a peak SSIM of 0.64, whereas peak PSNR is 21.81 and 22.87 respectively. This could be due to a large range of factors such as the the quality and uniformity of the DSLR images, the difference

**(a)** Digital **(b)** Film **(c)** MSE/VGG **(d)** Color/VGG/TV-Rel

**Fig. 8: Full Dataset Select Samples.** Outputs from select models on the full dataset. We see that the best performing model is MSE/VGG with resizing, which produces the best colour effect. We also see that the grain effect is not as strong as in the single image experiments.

**Table 5: Full Dataset Noise and Resizing.**
Comparison of MSE and MSE/VGG with and without noise and resizing.

| Loss | Noise | Resize | SSIM | PSNR | LPIPS | PieAPP |
|---|---|---|---|---|---|---|
| MSE/VGG | Yes | Yes | **0.64** | **22.87** | **0.27** | 1.90 |
| MSE/VGG | No | Yes | 0.63 | 22.85 | **0.27** | **1.84** |
| MSE/VGG | No | No | 0.58 | 20.47 | 0.50 | 3.67 |
| MSE | No | Yes | **0.63** | **22.20** | **0.35** | **2.31** |
| MSE | Yes | Yes | 0.59 | 20.87 | 0.40 | 3.29 |
| MSE | No | No | 0.58 | 20.47 | 0.50 | 3.67 |

**(a)** Digital **(b)** Film **(c)** MSE/VGG **(d)** MSE/VGG/TV-Rel **(e)** Color/VGG/TV-Rel

**Fig. 9: Full Dataset Grain Comparison.** Predictions for a patch of sky from select loss functions, all with noise. Similarly to the single image experiments, the models with losses that target some noise produce a more visually pleasing result. The best visual result is produced by Color/VGG/TV-Rel although this is still far from the true grain texture.

in model architecures or the difference in losses. However, perhaps the biggest reason is the greater complexity of the film effect.

In summary, we find that the best performing model is MSE/VGG with resizing, which produces the best metrics across the board, produces the best colour effect and often the most visually pleasing result too. The addition of noise has mixed effects but tends to help in noise production, and resizing the cropped patches during training significantly improves the learned colour transformation. The perceptual metrics better capture the output of noise by the model, and resizing allows the model to see a greater variety of areas of the image in a single patch. However, the full dataset is more complex and the models do not outperform the baseline as they did in the single image experiments. Our best model learns a good colour mapping, but the grain effect is not as strong as in the single image experiments, nor is it exactly the kind of grain observed.

## 5 Conclusion

In this work, we have explored the use of convolution neural networks in modeling the effect given by Cinestill800T film. We focused on the use of different loss functions as well as the addition of a noise channel to the input, and the use of random scales of patches during training. We find that a combination of MSE/VGG gives the best colour production which is significantly helped by resizing, and that the addition of a noise channel and another loss such as TV-Rel produces some grain.

Our contributions include the creation of a dataset of paired images taken with a film and digital camera, and all of our code in Pytorch, the details of which can be found in Section 7.1 in the appendix.

Our results are severely limited by the small size and great variety of our dataset, meaning that the model must learn a complex function with relatively little training data. This is evidenced by the lack of sign of halation ever being produced by our models, as they were simply not trained on enough patches containing halation. We also restricted ourselves to a pure deep learning approach, and did not explore the use of statistical models for film grain, halation, and colour hue.

## 6 Further Work

While we tried many combinations of hyperpamaters, we did not rigorously explore all options. Further work could systematically investigate the effect of kernel size, model size, patch size or the weightings for each loss function when combined. The dataset could be also narrowed focused to include only scenes of a similar nature to give the model an easier function to learn. This could be taken to the extreme for halation, where only patches containing some halation could be fed to the model. A natural extension would be to condition the model on features of the image such as dynamic range.

For grain production, an implicitly defined texture loss that more precisely captures the grain seen in films could be added, such as GCLM [29] or notably an adversarial loss as in [9]. Avenues we only briefly explored could be tested in more depth. This includes the paired auto-encoder architecture from [1] and the contextual bilateral loss from [31], both contained in our code. Lastly, the model could be combined with a non-ML approach to see if a hybrid model can produce better results.

## References


1. Afifi, M., Abuolaim, A.: Semi-supervised raw-to-raw mapping. CoRR **abs/2106.13883** (2021), https://arxiv.org/abs/2106.13883 3, 5, 15
2. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000) 4
3. Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. CoRR **abs/1605.00075** (2016), http://arxiv.org/abs/1605.00075 3
4. Deshpande, A., Rock, J., Forsyth, D.A.: Learning large-scale automatic image colorization. 2015 IEEE International Conference on Computer Vision (ICCV) pp. 567–575 (2015), https://api.semanticscholar.org/CorpusID:13195504 3
5. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks (2015) 2
6. Galerne, B., Raad, L., Lezama, J., Morel, J.M.: Scaling painting style transfer (2022) 3
7. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2414–2423 (2016). https://doi.org/10.1109/CVPR.2016.265 3
8. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization (2017) 3
9. Ignatov, A., Kobyshev, N., Vanhoey, K., Timofte, R., Gool, L.V.: Dslr-quality photos on mobile devices with deep convolutional networks. CoRR **abs/1704.02470** (2017), http://arxiv.org/abs/1704.02470 3, 5, 6, 12, 15
10. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks (2018) 2
11. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution (2016) 2, 3
12. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution (2016) 5
13. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation (2018) 2
14. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks (2019) 2
15. Kim, J., Lee, J.K., Lee, K.M.: Accurate image super-resolution using very deep convolutional networks (2016) 2
16. Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., Shi, W.: Photo-realistic single image super-resolution using a generative adversarial network (2017) 2
17. Liu, X., Wan, L., Qu, Y., Wong, T.T., Lin, S., Leung, C.S., Heng, P.A.: Intrinsic colorization. ACM Trans. Graph. **27**(5) (2008). https://doi.org/10.1145/1409060.1409105, https://doi.org/10.1145/1409060.1409105 3
18. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer (2017) 3
19. Mechrez, R., Talmi, I., Zelnik-Manor, L.: The contextual loss for image transformation with non-aligned data (2018) 2
20. Menon, S., Damian, A., Hu, S., Ravi, N., Rudin, C.: Pulse: Self-supervised photo upsampling via latent space exploration of generative models (2020) 2
21. Muja, M., Lowe, D.: Fast approximate nearest neighbors with automatic algorithm configuration. vol. 1, pp. 331–340 (01 2009) 4
22. Newson, A., Faraj, N., Galerne, B., Delon, J.: Realistic Film Grain Rendering. Image Processing On Line **7**, 165–183 (2017), https://doi.org/10.5201/ipol.2017.192 1

23. Prashnani, E., Cai, H., Mostofi, Y., Sen, P.: Pieapp: Perceptual image-error assessment through pairwise preference. CoRR **abs/1806.02067** (2018), http://arxiv.org/abs/1806.02067 9
24. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation (2015) 5
25. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: an efficient alternative to sift or surf. pp. 2564–2571 (11 2011). https://doi.org/10.1109/ICCV.2011.6126544 4
26. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015) 3, 6
27. Wang, X., Yu, K., Dong, C., Loy, C.C.: Recovering realistic texture in image super-resolution by deep spatial feature transform (2018) 2
28. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C.C., Qiao, Y., Tang, X.: Esrgan: Enhanced super-resolution generative adversarial networks (2018) 2
29. Wang, Y.D., Swietojanski, P., Armstrong, R.T., Mostaghimi, P.: Pixel co-occurence based loss metrics for super resolution texture recovery (2020), https://openreview.net/forum?id=rylrI1HtPr 15
30. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. CoRR **abs/1801.03924** (2018), http://arxiv.org/abs/1801.03924 9
31. Zhang, X.C., Chen, Q., Ng, R., Koltun, V.: Zoom to learn, learn to zoom. CoRR **abs/1905.05169** (2019), http://arxiv.org/abs/1905.05169 2, 15
32. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks (2020) 2

# 7 Appendix

## 7.1 Code Documentation

The code for this project is available at https://github.com/mikasenghaas/sillystill/tree/main. The code is written in Python and uses PyTorch, Lightning and Hydra for configuration. The code is structured as follows:

- `/src` contains the main code for the project, including the data processing, models, losses and training scripts.
  - `/data` contains all datasets and Lightning datamodules.
  - `/models` contains the PyTorch model classes and loss functions.
  - `/eval` contains wrapper classes for evaluation metrics.
  - `/utils` contains a variety of utility functions.
  - `train.py` is the main training script.
  - `dev.py` is a developer script for testing code without using Lightning or Hydra.
  - `preprocess.py` is a script for preprocessing the dataset.
  - `evaluate.py` is an evaluation script.
  - `unsplash.py` is a script for downloading images from Unsplash.
- `/data` contains our dataset
- `/configs` contains the Hydra configuration files for the experiments.
- `/notebooks` contains Jupyter notebooks for investigating the data processing, losses, models and evaluation.
- `/report` contains the Latex files for the report.

In order to train a model, the `src/train.py` script can be run with the desired configuration. You can specify a hydra configuration with flags or by editing the configuration files. The two experiments in this report can be run with the following commands:

```
python src/train.py experiment=single-image logger=none
python src/train.py experiment=full-data logger=none
```

To edit the loss functions or other parameters, you can edit the configuration file `configs/experiment/single-image.yaml`. The model will be trained, and inference will be run automatically, with results logged locally. If you want to log to weights and biases, you can remove the logger flag.

### 7.2 All Results

**Table 6: Single Image.** All Results

| Configuration | | | Evaluation | | | |
|---|---|---|---|---|---|---|
| Loss | Noise | Resized | SSIM | PSNR | LPIPS | PieAPP |
| MSE + VGG | No | Yes | **0.71** | **23.19** | 0.35 | 1.62 |
| MAE | Yes | No | 0.68 | 19.04 | 0.33 | 1.96 |
| Color + VGG + TV-Rel | No | No | 0.68 | 19.67 | 0.45 | 1.42 |
| Color + VGG | No | No | 0.68 | 20.07 | 0.45 | 1.61 |
| Color + VGG | Yes | No | 0.68 | 17.94 | 0.36 | 1.58 |
| Color + VGG + TV-Rel | Yes | No | 0.67 | 18.01 | 0.37 | **1.14** |
| MSE | No | Yes | 0.67 | 20.03 | 0.47 | 1.77 |
| MSE | Yes | No | 0.67 | 19.26 | 0.32 | 1.89 |
| MSE + VGG | Yes | No | 0.67 | 19.79 | 0.34 | 2.05 |
| MSE | Yes | Yes | 0.67 | 18.61 | 0.30 | 1.87 |
| MSE + TV-Rel + VGG | Yes | No | 0.66 | 20.14 | 0.30 | 1.84 |
| MSE + VGG | No | No | 0.66 | 18.26 | 0.48 | 2.53 |
| MAE | No | No | 0.65 | 18.61 | 0.50 | 2.12 |
| Color | No | No | 0.65 | 18.45 | 0.50 | 2.99 |
| MSE | No | No | 0.65 | 18.29 | 0.50 | 2.42 |
| **Baseline** | - | - | 0.64 | 16.97 | 0.36 | 3.79 |
| Color | Yes | No | 0.63 | 15.66 | 0.43 | 2.50 |
| TV-Rel | Yes | No | 0.27 | 11.11 | **0.27** | 2.20 |
| TV-Rel | No | No | 0.17 | 9.84 | 0.60 | 4.84 |
| VGG | No | No | 0.16 | 9.10 | 0.62 | 4.50 |
| VGG | Yes | No | 0.06 | 8.71 | 0.64 | 5.11 |

**Table 7: Full Dataset.** All results

| Configuration | | | Evaluation | | | |
|---|---|---|---|---|---|---|
| Loss | Resize | Noise | SSIM | PSNR | LPIPS | PieAPP |
| **Baseline** | - | - | **0.64** | 21.68 | **0.26** | 1.96 |
| MSE + VGG | Yes | Yes | **0.64** | **22.87** | 0.27 | 1.90 |
| MSE + VGG | Yes | No | 0.63 | 22.85 | 0.27 | **1.84** |
| MSE | Yes | No | 0.63 | 22.20 | 0.35 | 2.31 |
| MSE | Yes | No | 0.59 | 19.68 | 0.48 | 3.38 |
| Color + VGG + TV-Rel | No | Yes | 0.59 | 21.42 | 0.41 | 3.07 |
| MSE | No | No | 0.58 | 20.47 | 0.50 | 3.67 |
| MSE + VGG | No | No | 0.58 | 20.47 | 0.50 | 3.67 |
| MSE + VGG + TV-Rel | No | Yes | 0.58 | 20.43 | 0.46 | 3.54 |