Automatic colorization of videos

Tanvi Sahay
University of Massachusetts Amherst
tsahay@cs.umass.edu

Ashutosh Choudhary
University of Massachusetts Amherst
ashutoshchou@cs.umass.edu

Abstract

Realistic colorization of videos has been of great interest to the artistic community, primarily for restoring historical color films and colorizing legacy videos. In this project, we experimented with several methods in order to automatically colorize videos on a frame-by-frame basis. We focused on rectifying two primary issues encountered with video colorizations : lack of color consistency between subsequent frames and desaturated colorization of individual frames. We used an LSTM to encode the sequential information of videos and thus maintain color consistency between successive frames. We used a class-rebalancing loss to reweight color predictions on the basis of their rarity. We evaluated out results using average per-pixel RMSE over all frames in a single video and also set up a colorization "Turing Test" to determine which models gave the most realistic colorization.

1. Introduction

The task of automatically hallucinating colors for grayscale images has seen a lot of interest in recent years in the Computer Vision community[7][6],[2]. Not only is it interesting to see how our existing deep learning models perform on this task but having a good colorizer can also allow us to perform colorization on legacy images and perform downstream tasks such as object detection on these images using networks trained on color images. A compelling extension of automatically colorizing images is video colorization, which has been a popular part of the motion pictures as a means to modernize black-and-white movies or provide an artistic visual effect to them, to restore original color movies and videos and to integrate originally black-and-white videos into modern day color films. In this project, we attempted to automatically colorize videos on a frame by frame basis by modifying the recent popular architectures for image colorization.

We compared two baseline image colorization architectures([7],[6]) and analyzed their performance on video colorization. Results from both architectures showed that consecutive frames had no consistency between their color predictions, despite the fact that the frames were very close to each other in terms of the objects present in each frame. This shows up as a '*flickering*' and is a limitation of the architecture. This limitation stems from the fact that the model has no information of the frames it has seen in the past while predicting the color values of a particular frame. This flickering gets aggravated when To account for this sequential nature of video frames, we took inspiration from [4] and implemented an LSTM on top of the present CNN based architectures. Additionally, in order to avoid getting desaturated colorization of individual frames, we borrowed from [7] and implemented a class-rebalance loss in order to achieve brighter and more realistic colorizations per frame.

In the following sections, we describe both baseline architectures in detail, along with the sequence architecture implemented by us. Section 2 delves into some of the relevant previous works, section 3 describes our sequential framework and section 4 provides details of the experiments and their results.

2. Prior Work

2.1. Deep Colorization [2]

In this paper, Cheng et.al. presented the first fully automated technique of colorizing images in 2016, by treating the image colorization as a regression problem to predict the ab channel values given the lightness channel and using deep neural networks to solve it. They propose an adaptive image clustering technique to group reference input images and train a deep neural network for each cluster. Given a pair of reference images $\Lambda = \overrightarrow{\mathrm{G}}, \overrightarrow{\mathrm{C}}$, where $\overrightarrow{\mathrm{G}}$ corresponds to a grayscale image and $\overrightarrow{\mathrm{C}}$ corresponds to its colored counterpart, each DNN is trained on feature descriptors for pixels in the grayscale image to predict the chrominance values of color images as outputs. Given this formulation, they try to minimize the least squares problem:

$$argmin_{\theta \in \Upsilon} \sum_{p=1}^{n} ||F(\theta, x_p) - c_p||^2 \qquad (1)$$

where F is the function that maps input feature descriptor of the grayscale image to chrominance values of the color image, θ are parameters of this function, x_p is the feature descriptors extracted from pixel p, c_p is the chrominance value of the corresponding color pixel, n is the number of training pixels sampled from input and Υ is the function space of $F(\theta, x_p)$. Contradictory to their approach, we model the colorization problem as a classification problem, similar to the more recent work. However, similar to their approach, we transform our images to the Lab space and predict ab values given lightness values per pixel.

2.2. Colorful Image Colorization [7]

In this paper, Zhang et.al. formulated the image colorization problem as a classification problem and used a VGG-styled network modified with extra layers for added depth and dilated convolutions to hallucinate realistic looking colorizations of grayscale images. They predict a distribution of colors for each pixel and re-weight the loss at training time to put more emphasis on rare colors. Their network optimizes the multinomial cross-entropy loss:

$$L_{cl}(\hat{\mathbf{Z}}, \mathbf{Z}) = -\sum_{h,w} v(\mathbf{Z}_{h,w}) \sum_{q} \mathbf{Z}_{h,w,q} log(\hat{\mathbf{Z}_{h,w,q}}) \quad (2)$$

where $v(.)$ is the weighting factor used to rebalance the loss, $\hat{Z}$ is the probability distribution over possible colors, Z is the vector received by vectorizing ground truth color using a soft encoding scheme, against which the predicted distribution is compared. Subscripts h, w and q denote the dimensions of the image, where h and w are height and width respectively and q is the number of bins in which ab values have been quantized.

To rebalance the loss to account for class rarity, each pixel is weighted by the weight corresponding to its closest ab bin. In this project, we build upon their original architecture and make use of the class-rebalance loss to avoid desaturation of individual frames.

2.3. Learning representations for automatic colorization [6]

In 2017, Larsson et.al. proposed a hypercolumn based architecture to predict per-pixel histogram of colors over a set of color bins instead of a single color to account for scene elements that can draw from several colors. They framed the colorization problem as learning a function $f : X \rightarrow Y$, where X is the input space, Y is the output space and the function is implemented as a VGG network. The experimented with two color spaces - Lab and HSV where the V and S channels are modified using a color bicone such that: $V = L + \frac{C}{2}$ and $S = \frac{C}{V}$, where L and C are lightness and chroma values respectively and each channel space binned separately into 32 bins.

Given these color spaces, they defined a loss over the predicted color distributions as:

$$L_{hist}(\mathbf{x}, \mathbf{y}) = D_{KL}(\mathbf{y}||f(\mathbf{x})) \quad (3)$$

where $Y = [0, 1]^K$ described a histogram over K bins and D_{KL}, is the KL-divergence. To account for instability of Hue as Chroma approaches 0, they also added a sample weight to their loss:

$$L_{hue/chroma}(\mathbf{x}, \mathbf{y}) \quad (4)$$

$$= D_{KL}(\mathbf{y}_C||f_C(\mathbf{x})) + \lambda_H y_C D_{KL}(\mathbf{y}_H||f_H(\mathbf{x})) \quad (5)$$

In this project, we used their pre-trained model on video frames and compared the results to our model as well as the zhang model.

2.4. Long-term Recurrent Convolutional Networks for Visual Recognition and Description [4]

In this paper, Donahue et. al. propose the LRCN(Long Recurrent Convolutions Network) architecture that combines a deep hierarchical feature extractor like CNN with a model capable of long-range temporal recursion like LSTM to process sequential data. The model works by passing each image through a feature transformation $\phi_V(.)$, generally a CNN to produce fixed length feature representations for each input image ($\phi_V(x_t)$) where x_t is the t^{th} image, V is the set of parameters of the CNN and ϕ is the feature transformation function. These features are then passed into a recurrent neural network module (RNN or LSTM) that predicts a distribution P(y_t) over outcomes $y_t \in C$ at time t by passing its outputs $z_t \in \mathbb{R}^{d_z}$ through a linear prediction layer $\hat{y_t} = W_z Z_t + b_z$, where W and b are learned parameters and finally applying the softmax function on $\hat{y^t}$:

$$P(y_t = c) = softmax(\hat{y_t}) = \frac{exp(\hat{y_t}, c)}{\sum_{c' \in C} exp(\hat{y_t}, c')} \quad (6)$$

They tested the architecture in three setting: sequential input and static output, static input and sequential output and sequential input and sequential output. In this project, we have build used the architecture similar to the one they propose for the first setting. However, instead of taking an average over all time steps, we make use of the final time step output to compare with the ground truth.

3. Data

For this project, video data was extracted from the YouTube-8M [1], which is a large-scale labeled video dataset that provides more than 3 million videos , each 120-500 seconds long, divided into 4716 categories. Of this, we chose the category 'movieclips', that contained 1533 videos of varying length. Details for the videos are provided in table 1.

Table 1. Dataset

Parameter	Value
input video formats	.mp4, .webm
output video formats	.mp4
input frame format	.png
frames per second	25 fps
bitrate	330k bps
audio stream frequency	441k Hz
original resolution	720 x 1080
Models based on Zhang et al. input frame cropping	224 x 224
Models based on Larson et al. input frame cropping	256 x 256
number of training frames	207360
number of val frames	25920
number of test frames	25920
LSTM sequence length	6

Out of the 1533 videos available, 60 were randomly chosen and divided into train, val and test categories according to the ratio 80:10:10. The final models were trained on a total of 48 videos and tested on 6 videos, with the remaining 6 used as a validation set.

3.1. Preprocessing

Each video in the train, test and val set was divided into frames according to the fps value given in table 1. For a standard 2 minute video containing 25 frames per second, 3000 frames were extracted. Once the color frames were extracted, train and val frames were used to train the experimental models, within which lightness and ab channels were separated. The lightness channel was used as input to the models and ab channel values were as the ground truth predictions. For testing the models, lightness channels for test frames were extracted and these grayscale frames were given as test inputs.

4. Technical Approach

The underlying task of image or video colorization is not to necessarily recover the actual ground truth color of the image or video, rather it is to produce a plausible colorization that could potentially fool a human test subject [7]. Although, this task may not be the most popular way in which image colorization tasks have been defined in the past, it is a more achievable one since it models enough of the statistical dependencies between the semantics and textures of grayscale images and their color version. Thus, we propose an architecture that both predicts more realistic colorizations and maintains the sequential structure of input frames.

4.1. Architecture

We train a Long-term Recurrent Convolutional Network[4] mapping a grayscale input to a distribution over quantized color value outputs using the architecture in figure 1. We temper the loss from CNN by sequential information provided by the LSTM part of the architecture as weighted sum of losses 1.

The network consists of 7 conv layers each consisting of 2 or 3 repeated conv and ReLU layers, followed by a batchNorm[5] layer. The last conv layer is followed by a deconv layer used to upsample the image to obtain dimensions suitable for quantized classification loss of CNN. The network does not have any pool layers. All changes in resolution are obtained by using downsampling or upsampling convolution blocks. The architecture till the seventh conv layer resembles a modified VGG-16 architecture. The modifications in VGG-16 and the objective function/classification loss from CNN are the ones presented in [7]. The logits of the CNN network serve as function approximators or vector representations of the frames used as input to the LSTM network. Batching without shuffle helps retain the sequence of frames for LSTM network where the labels are the sequence vector representations of the frame color labels used for CNN network. Hence, the LSTM behaves as a cross-channel auto-encoder.

4.2. Objective Function

Given the lightness channel L of an image, our model tries to predict the corresponding a and b color channels of the image in the CIE Lab colorspace. Since color prediction is described as multimodal classification in [7], given an input channel $\mathbf{X} \in \mathcal{R}^{HxWx1}$ our CNN objective is to learn a mapping $\hat{\mathbf{Y}} = \mathcal{F}(\mathbf{X})$ to the two associated color channels $\mathbf{Y} \in \mathcal{R}^{HxWx2}$, where H, W are image dimensions after preprocessing step. We quantize the ab output space into bins with grid size 10 and $Q = 313$ which is the number of quantized bins in ab space. A function $Z = \mathcal{H}_{gt}^{-1}\mathbf{Y}$, which uses a soft-encoding scheme[7], converts ground truth color $\mathbf{Y}$ to vector $\mathbf{Z}$. We compare the CNN predicted quantized colors $\hat{\mathbf{Z}}$ to $\mathbf{Z}$ to obtain the multimodal cross entropy loss L_{cl}

4.3. Class rebalancing

Image background such as clouds, pavement, dirt, walls are statistically more probable to appear in a dataset. Due to this, the distribution of ab value is strongly biased towards low ab values. Thus, desaturated values in a natural image are orders of magnitude higher than for saturated values. If not accounted for, this would lead to a loss function biased towards desaturated ab values. The class-imbalance problem in this classification task is resolved by re-weighting the loss of each pixel at train time based on pixel color rarity.

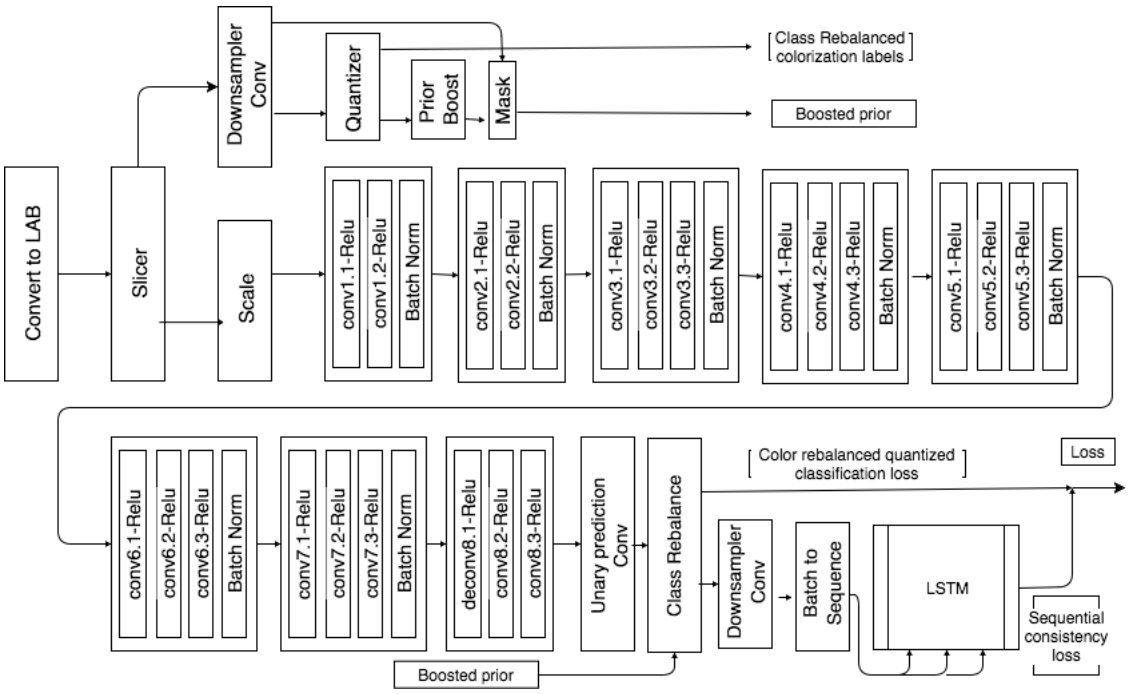

Figure 1. Model Architecture

This re-weighting can be achieved by re-sampling the training space such that each pixel is weighed by factor $\mathbf{w} \in \mathcal{R}^Q$ based on its closest ab bin.

$$v(\mathbf{Z}_{h,w}) = w_{q*}\,, where\ \ q* = argmax_q \mathbf{Z}_{h,w,q} \tag{7}$$

$$\mathbf{w} \propto ((1-\lambda)\tilde{\mathbf{p}} + \frac{\lambda}{Q})^{-1} \tag{8}$$

$$\mathcal{E}[\mathbf{w}] = \sum_q \tilde{\mathbf{p}}_q \mathbf{w}_q = 1 \tag{9}$$

Smoothening of distribution $\tilde{\mathbf{p}} \in \Delta^Q$ can be achieved by estimating the empirical probability of colors in the quantized ab space $\mathbf{p} \in \Delta^Q$ from the entire training set and smooth the distribution with a Gaussian kernel $\mathbf{G}_\sigma$. The distribution is then mixed with a uniform distribution with weight $\lambda \in [0,1]$, inversed and normalized such that the weighting factor is 1 on expectation.

4.4. Loss function

Using equations [7,8,9] and the definition of cross-entropy, multinomial cross entropy loss with class rebalancing for the CNN subgraph is defined as:

$$L_{cl}(\hat{\mathbf{Z}}, \mathbf{Z}) = -\sum_{h,w} v(\mathbf{Z}_{h,w}) \sum_q \mathbf{Z}_{h,w,q} log(\hat{\mathbf{Z}_{h,w,q}}) \tag{10}$$

The autoencoder part (LSTM), requires re-sampling of labels $\mathbf{Y}$ using a deconv layer, to obtain $\mathbf{Y}_{seq}$. The sequence based multimodal extracted-feature classification loss is basically a softmax over sequence objective function.

$$\mathbf{L}_{seq} = -\sum_j \mathbf{y}_j log(\frac{e^{\mathbf{s}_j}}{\sum_k^q e^{\mathbf{s}_k}}) \tag{11}$$

where $\mathbf{y}_j \in \mathbf{Y}_{seq}$. The final loss function is a weighted sum of the two loss functions described. This choice of loss function is based on the idea that the primary loss which should impact colorization to a large extent should be $\mathbf{L}_{cl}$. $\mathbf{L}_{seq}$ should impact the overall loss if there is a large deviation of current frame colorization from the previous one. This loss is suitable for our problem because with 24fps frame rate from video produces highly correlated frames. The overall loss therefore is:

$$\mathbf{L} = \beta \mathbf{L}_{cl} + (1-\beta)\mathbf{L}_{seq} \tag{12}$$

The choice of β is critical in the experiments and could be learned using hyper-parameter optimization. In the interest of computational runtime, we babysit the process of choosing β and chosen a value of 0.8 .

5. Experimentation and Results

We experimented with 5 different models and compared their results on the basis of root mean square error between the predicted and ground truth colors. We also set up a turning test to determine videos colorized by which method were preferred more by human evaluators.

Table 2. Optimal Hyperparameters for each experimental model

Model	base lr	lr policy	step value	γ	max iter	momentum	weight decay	optimizer
larsson et.al.	0.001	"multistep"	50000 65000 80000	0.1	1000000	0.9	0.0005	Adam
zhang et.al.	3.16e-5	"step"	43000	0.316	450000	0.9, 0.99	0.001	Adam
zhang retrained	3.16e-5	"step"	43000	0.316	50000	0.9, 0.99	0.001	Adam
LRCN(our model)	3.16e-5	"step"	43000	0.316	50000	0.9, 0.99	0.001	Adam
Ensemble(our model)	Same values as in larsson et.al. and zhang et.al.							

5.1. Experimental Models

The models experimented with have been briefly described below. Hyperparameters used for training each of these models have been provided in table 2.

5.1.1 zhang et.al.

We downloaded the pre-trained model provided by zhang et.al. and tested our graycsale frames using this model. The model was trained on 1,000,000 color images taken from ImageNet[3] for 450000 iterations and each image was cropped to the shape 224x224.

5.1.2 larsson et.al.

We also downloaded the pre-trained model provided by larsson et.al. and tested it on our grayscale video frames. This model was trained on 1.2 million image net training set images and each image was cropped to at most 256 pixels in the smaller dimension.

5.1.3 zhang et.al. retrained

Since the original zhang model was trained on ImageNet, we fine-tuned the weights of the original model end-to-end using the 'movieclips' training data extracted by us. We tested the grayscale images on this retrained model as well.

5.1.4 Ensemble

We implemented an ensemble of outputs extracted from the pre-train zhang and larsson models as well. In order to compute the ensemble, we averaged the values predicted by both models for each pixel in the image.

5.1.5 LRCN

Finally, we tested the grayscale images with our LRCN model. We used the same hyperparameters as the ones specified in zhang's original paper, including cropping the images to the size 224x224 and trained the model for 50000 iterations. We initialized the CNN with the pre-trained zhang model and fine tuned these weights while learning the LSTM parameters from scratch.

5.2. Evaluation Metrics

5.2.1 RMSE

To evaluate how well each algorithm predicted the colors of each pixel, we took the pixel wise root mean square error between each predicted frame of each video and its corresponding ground truth color frame. For each frame, we averaged the error per pixel and finally, averaged the frame rmse for all the frames for each video to obtain a single value.

5.2.2 Colorization Turing Test (CTT)

While RMSE gives an estimate how similar to the original ground truth our predictions are, it is not always the final aim of a colorization task. Another way of evaluating colors is on the basis of realistic they look to a human evaluator. To evaluate this, we set up a colorization turing test where we asked 10 people to score each of our predicted colorized videos in the range of 1 to 5 on the basis of how realistic they looked.

5.3. Results

Results obtained for both evaluation metrics are shown in table 3. Figure 3 shows sample frames extracted from each experimental model along with the corresponding original frames.

Table 3. Results for each experimental model

	Per-pixel RMSE	CTT scores
larsson et.al.	7.277	3.4
zhang et.al.	8.8908	4.2
zhang retrained	6.54869	3.2
LRCN(our model)	6.8068	3.0
Ensemble(our model)	7.2878	4.5

From figure 2, it can be seen that the ensemble model generates the most realistic looking pictures while the col-

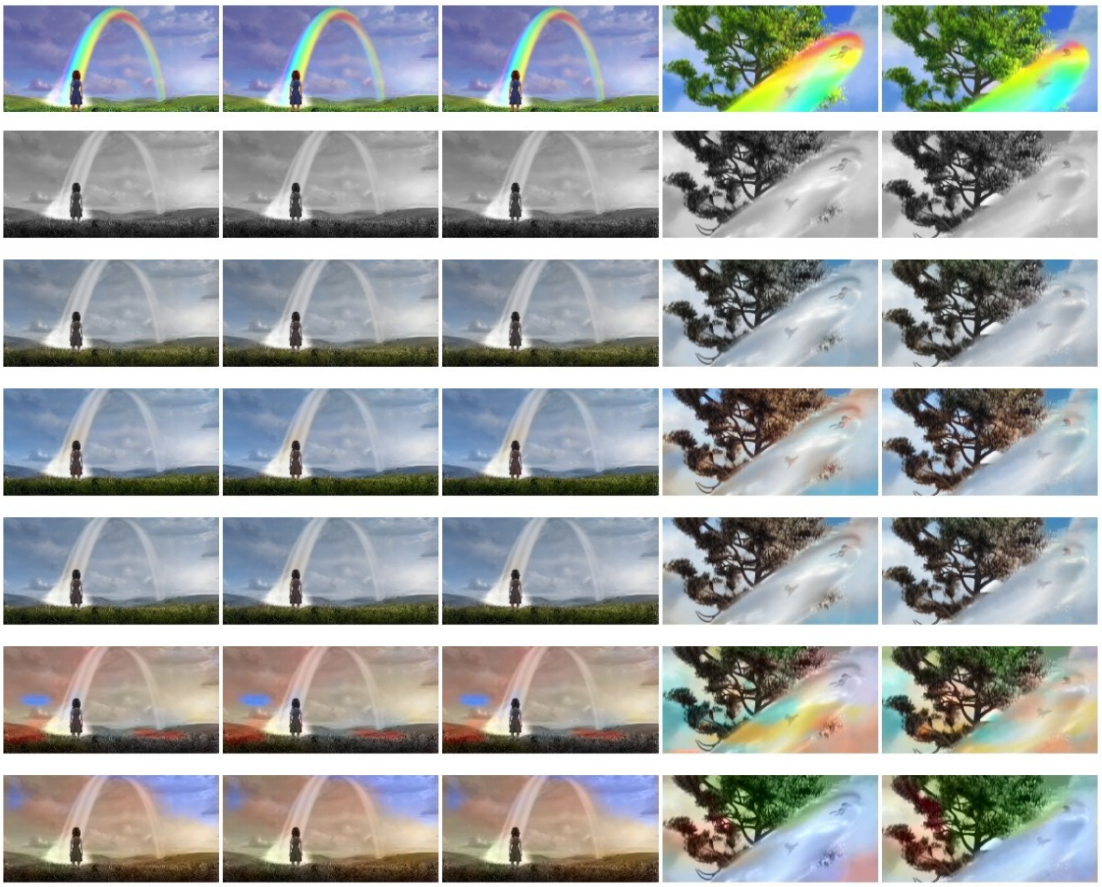

Figure 2. Sample frames extracted from each experimental model. 1) Original color frames. 2) is the grayscale input, 3) extracted from larsson et.al. 4) extracted from zhang et.al. 5) extracted from Ensemble model 6) extracted from retrained zhang and 6) extracted from LRCN model.

ors predicted by larsson look too desaturated and those predicted by zhang look too bright. From the images it can also be seen that for the LRCN model, subsequent frames have similar colors, and that the frames have more realistic and bright colors than the zhang or larsson model(particularly the sky and tree).

Figure 3 shows a series of frames for which the LRCN model did not perform well. However, the ensemble model still performed remarkably well for this sequence as well.

6. Conclusion

In this project, we experimented with a long-term convolution recurrent neural network architecture for the purpose of automatically colorizing grayscale videos and compared their results with standard image colorization models like zhang et.al., larsson et.al. and their ensemble. We observed that while larsson's model gave desaturated values that had low RMS error, zhang's predictions were brighter and more realistic, though further away from the ground truth, resulting in high RMS error. The Ensemble model compensated for the brightness of zhang's prediction and avoided the desaturation caused by larsson's predictions, thus giving the best resuls. We also observed that while the LRCN learned to maintain color between sequences of frames, the resultant predictions were very different from the ground truth, thus resulting in high RMSE values. We attribute this behavior to the sequential input single output architecture used by us and believe that the sequential input-output architecture would perform well on this task.

Figure 3. Sample frames extracted from each experimental model. 1) Original color frames. 2) is the grayscale input, 3) extracted from larsson et.al. 4) extracted from zhang et.al. 5) extracted from Ensemble model 6) extracted from retrained zhang and 6) extracted from LRCN model.

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *CoRR*, abs/1609.08675, 2016.

[2] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. *CoRR*, abs/1605.00075, 2016.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[4] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389, 2014.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[6] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. *CoRR*, abs/1603.06668, 2016.

[7] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016.