Internship Progress Report

Le-Vu Tran

August 19, 2018

Contents

Chapter 1

Week 10 - August 9 2018

1.1 Introduction

I have spent the majority of my time this week struggling with Long - Short Term Memory network, seeking for applicable video classification methods and figuring out what made an image memorable.

1.2 Video classification methods

Matt Harvey's paper[62] was about introducing some appropriate video classification methods. The author compared 5 different methods on UCF-101[49] to come up with the final results and his source code could be found here on his GitHub page. Here was the summary of 5 video classification methods introduced by the author.

Classify one frame at a time with a Convolutional Neural Network. Using this method meant we had ignored the temporal features of video and attempt to classify each video by looking at just a single frame. The most recommended technique for this method was using transfer learning to retrain a pre-trained Neural Network on different datasets such as ImageNet. We literally looked at each frame independently, and classified the entire video based solely on that one frame rather than looking at all the frames and doing any sort of averaging or max-ing. After training on UFC101, the author got the final test accuracy of 65% top 1.

Use a time-distributed Convolutional Neural Network, passing the features to an Recurrent Neural Network, in one network. The author suggested first using the Kera's TimeDistributed wrapper distribute layers of Convolutional Neural Network across an extra dimension - time; secondly, forwarding this wrapper through our Recurrent Neural Network. This model must be trained from scratch. After training, the author got the final test accuracy of 20%, which was much far from the baseline above.

Use a 3D Convolutional Neural Network. This seemed to be a reasonable method because 3D ConvsNet applied convolutions (and max pooling) in the 3D space, where the third dimension in our case was time. The author produced a network called C3D that achieved 52.8% accuracy on UFC101. After training, the author got the final test accuracy of 28%, which was also far from our baseline.

Extract features with a Convolutional Neural Network, pass the sequence to a separate Recurrent Neural Network. First, we ran every frame from every video through Inception (or some other networks), saving the output from the final pool layer of the network (feature). So we effectively chop off the top classification part of the network so that we ended up with a 2,048-d vector of features that we couls pass to our RNN. Second, we converted those extracted features into sequences of extracted features. The author stated the this model achieved better than the CNN-only results. The concretely accuracy number was 74%, which was significantly higher than our baseline.

Extract features from each frame with a Convolutional Network Network and pass the sequence to an Multi-Layer Perceptron. The author applied the same Convolutional Neural Network extraction process as in the previous method, but instead of sending each piece of the sequence to an Recurrent Neural Network, we would flatten the sequence and pass the new ***[40,***

2048] input vector into a fully connected network (a multilayer perceptron). After training, the author got the final test accuracy of 70%, which was higher than our baseline but lower than the previous method.

1.3 What makes an image memorable?

1.3.1 Introduction

After getting stuck in trying to improve the accuracy of my network architecture, I started researching which factors made an image or video memorable. I found a relevant paper in CVPR 2011 about this article which was ***What makes an image memorable?***[63].

Among the many reasons why an image might be remembered by a viewer, the authors investigated first the following factors: color, simple image features, object statistics, object semantics, and scene semantics.

1.3.2 Color and simple image features

The authors proposed a figure that demonstrate the correlation between memorability and basic pixel statistics. Mean hue was weakly predictive of memory: as mean hue transitions from red to green to blue to purple, memorability tended to go down (ρ = -0.16). This correlation might be due to blue and green outdoor landscapes being remembered less frequently than more warmly colored human faces and indoor scenes. Mean saturation and mean value, on the other hand exhibited even weaker correlations with memorability.

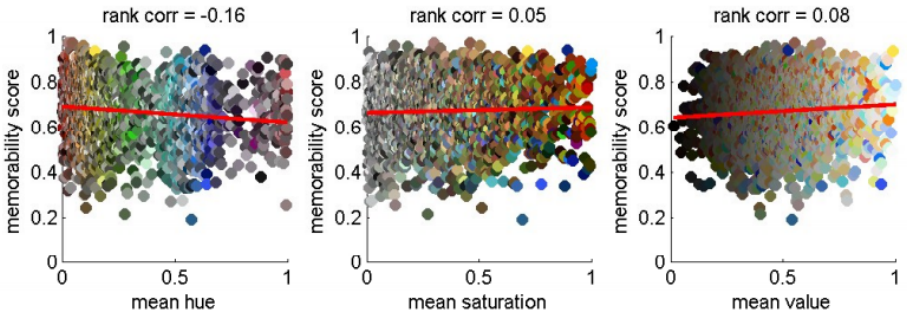

Figure 1.1: Correlation between Simple image features and Memorability.

These findings concorded with other work that had shown that perceptual features were not retained in long term visual memory[64]. In order to make useful predictions, more descriptive features were likely necessary.

1.3.3 Object statistics

Using LabelMe[65], each image in target set was segmented into object regions and each of these segments was given an object class label by a human user (e.g "person," "mountain," "stethoscope"). In this section, the authors quantified the degree to which the data could be explained by non-semantic

object statistics. The authors found that none of these statistics above made good predictions on their own.

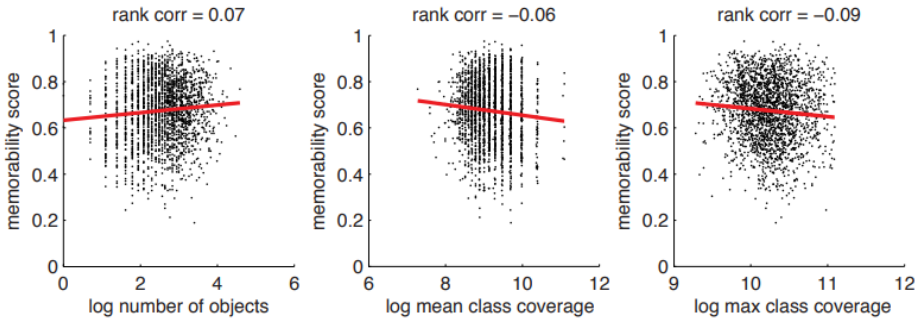

Figure 1.2: Correlation between Simple Object Statistics and Memorability 1.

Simple object statistics (log number of objects, log mean pixel coverage over present object classes, and log max pixel coverage over object classes) did not correlate strongly with memorability (ρ = 0.07, -0.06, and -0.09 respectively).

Beside that the authors also marginalized across classes to generate histograms of Object Counts, Object Areas and Multiscale Object Areas. And these factors' correlation were even smaller than those of the color and simple image features.

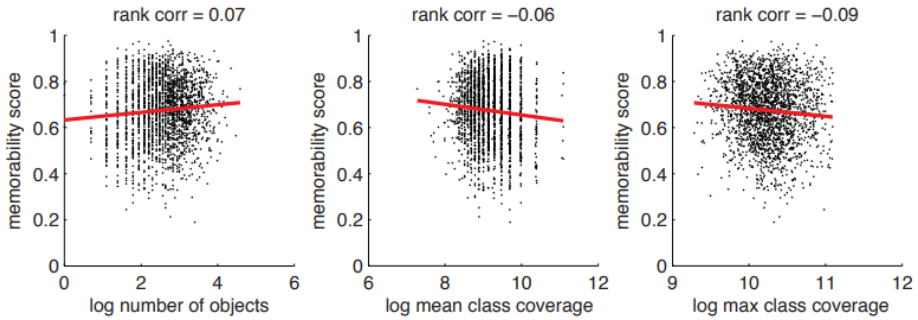

Figure 1.3: Correlation between Simple Object Statistics and Memorability 2.

1.3.4 Object and scene semantics

As demonstrated above, objects without semantics were not effective at predicting memorability. Using regression on the entire joint (object class, statistic), the authors proposed a comparision table of predicted versus measured memorabilities.

	Object Counts	Object Areas	Multiscale Object Areas	Object Label Presences	Labeled Object Counts	Labeled Object Areas	Labeled Multiscale Object Areas	Scene Category	Objects and Scenes	Other Humans
Top 20	68%	67%	73%	84%	82%	84%	84%	81%	85%	86%
Top 100	68%	68%	73%	79%	79%	82%	82%	78%	82%	84%
Bottom 100	67%	64%	64%	57%	57%	56%	56%	57%	55%	47%
Bottom 20	67%	63%	65%	55%	54%	53%	52%	55%	53%	40%
ρ	0.05	0.05	0.20	0.43	0.44	0.47	0.48	0.37	0.50	0.75

Figure 1.4: Comparision of Predicted versus Measured Memorabilities.

Even the Object Label Presences alone, which simply convey a set of semantic labels and otherwise did not describe anything about the pixels in an image, performed well above the authors' best unlabeled object statistic, Multiscale Object Areas ($\rho = 0.43$ and 0.20 respectively). Moreover, Scene Category, which just gave a single label per image, appeared to summarize much of what makes an image memorable ($\rho = 0.37$), and the best method was to combine both object and scene semantic information ($\rho = 0.50$). These performances supported the idea that object and scene semantics were a primary substrate of memorability.

1.3.5 Visualizing what makes an image memorable

Since object content appears to be important in determining whether or not an image would be remembered, the authors further investigated the contribution of objects by visualizing object-based memory maps for each image.

This visualization gave a sense of how objects contribute to the memorability of particular images. The authors additionally interested in which objects were important across all images so they estimated an object's overall contribution as its contribution per image, calculated as above, averaged across all test set images in which it appeared with substantial size. The authors sorted objects into an intuitive ordering: people, interiors, foregrounds, and human-scale objects tend to contribute positively to memorability; exteriors, wide angle vistas, backgrounds, and natural scenes tend to contribute negatively to memorability.

Figure 1.5: Objects sorted by their predicted impact on memorability.

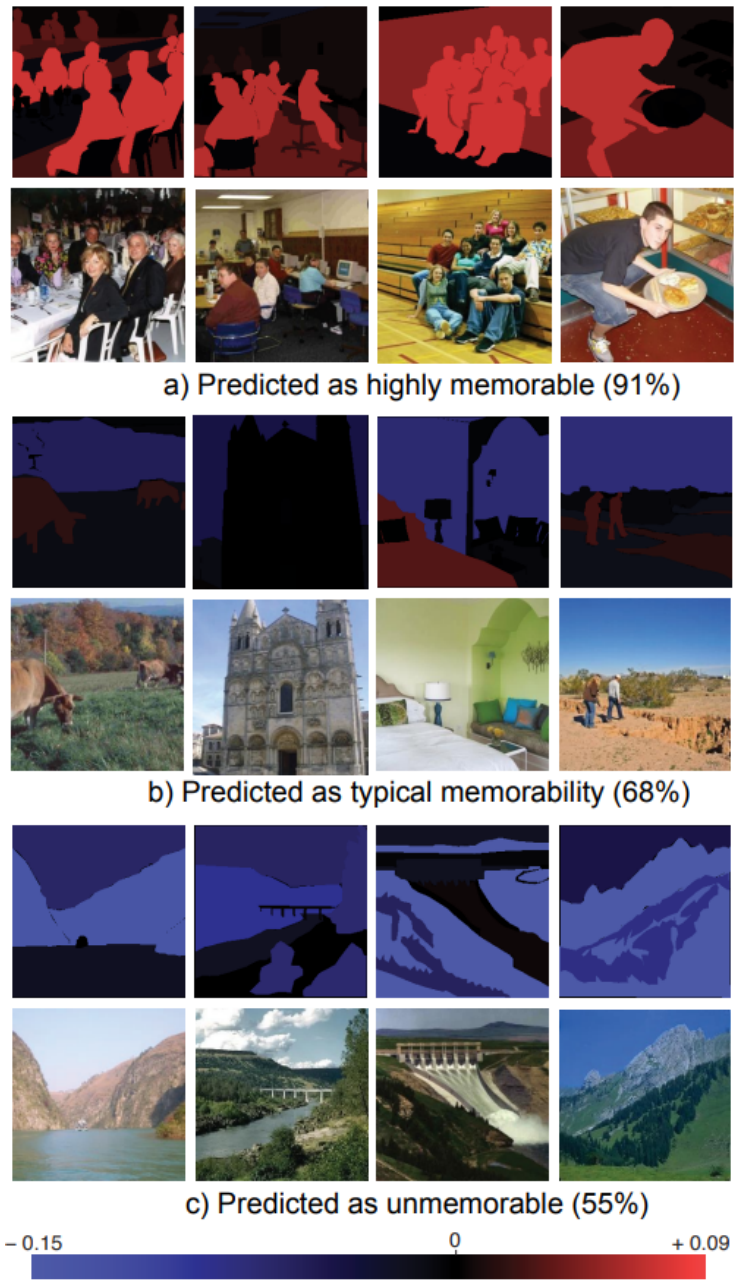

Figure 1.6: Visualization of how each object contributes to the memorability of sample images spanning a range of memorability predictions. In red were objects that contribute to higher predicted memorability and in blue were objects that contribute to lower predicted memorability. Brightness was proportional to the magnitude of the contribution.

1.4 PyTorch LSTM Classification Network

1.4.1 Architecture

I misunderstood between the concept of number of layers and number of Long - Short Term Memory (LSTM) cells so the training time was nearly 4 times longer than normal. For more details, the number of LSTM cells that PyTorch used was actually the quantity of sequences we feed into the LSTM network at once. So in my case it was 8 (because each video had 8 frames, and I feed all 8 frames once). The number of layers term in PyTorch meant how many times I wanted to stack the 8-LSTM-cell sequence to form a $\boldsymbol{8 \times num\_layers}$ grid of LSTM cells.

Each LSTM cell had multiple LSTM units, and the quantity of units was defined by ***hidden_size*** parameter. The figure below described correctly the Long - Short Term Memory model architecture in PyTorch.

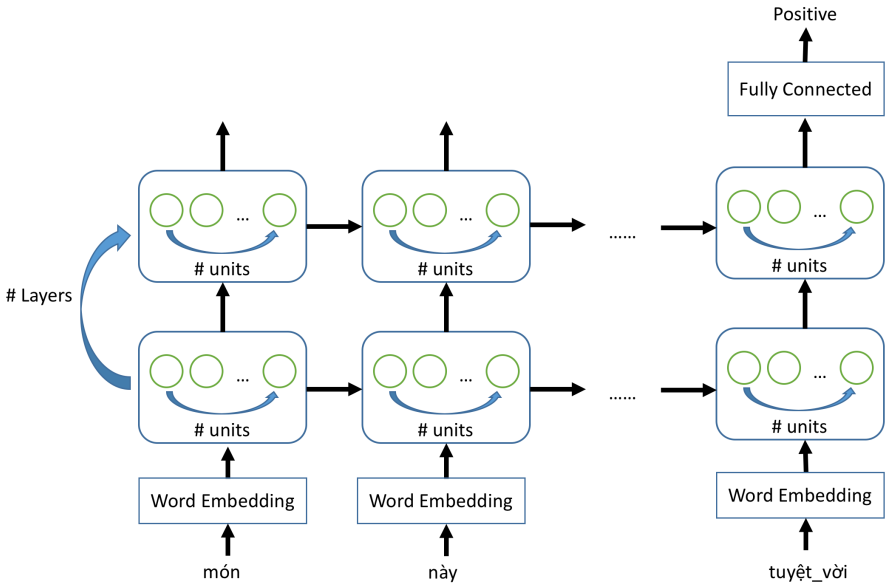

Figure 1.7: Long - Short Term Memory Network Architecture.

The smallest unit in the network above was the LSTM unit, multiple LSTM units formed one LSTM cell, each cell took responsibility for one input feature (multiple input features formed the input sequence). Multiple LSTM cells formed one LSTM layer. Stacking multiple LSTM layers formed one Stacked LSTM network.

1.4.2 Training on Dev-Set (extracted by ResNet50)

Strategy

The purpose of this challenge was to calculate the memorability score for each video, and this calculation was so ambiguous so I broke this problem into the classification problem of ***11*** categories. So after getting the classify results, I would devide them by 10 to get numbers which were similar to the scores

required by the challenge. My strategy was to splitted the provided dev-set for this challenge into two parts, since the dev-set had 8000 videos, I picked ***6000*** videos for training and ***2000*** for testing.

Version 1

My network architecture contained 2 parts; the first part was the Long - Short Term Memory part, took the sequence of input and gave a sequence of output; the second part was the Fully-connected layer acted as the classifier. I used a sequence of 8 LSTM cells took in input of size *[8, 2048]* and lately produced output of size ***[8, 8]***. Next, I flattened that array and passed it through the Fully-connected Layer to finally got the classification score for *11* classes. There was a figure below to demonstrate exactly how my architecture was.

I trained this architecture with ***200*** epochs and at the learning rate of ***1e-1***. But the result was not impressive to me at all. I think because the learning rate was too high for this kind of network. The loss and accuracy were completely turbulent over time.

The maximum accuracy was ***26%*** and the minimum value of loss was ***74*** but those values were not reliable.

This architecture took me ***3 hours*** to train 200 epochs (***54s*** per epoch).

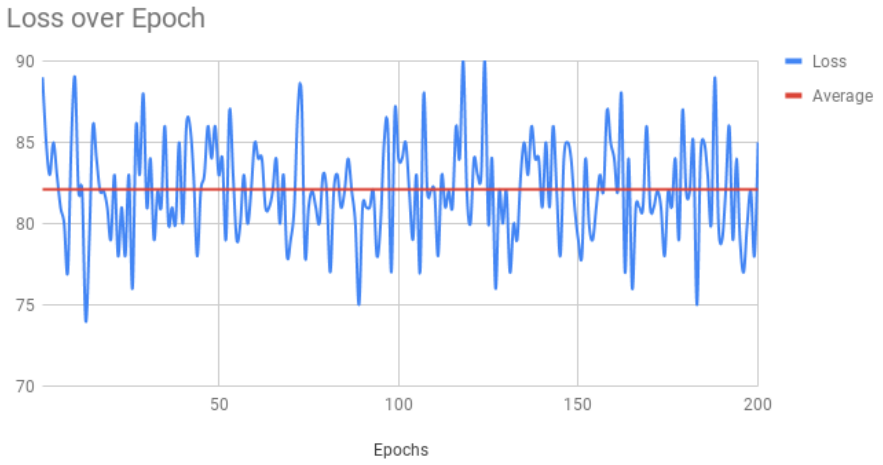

Figure 1.8: Loss over Epoch.

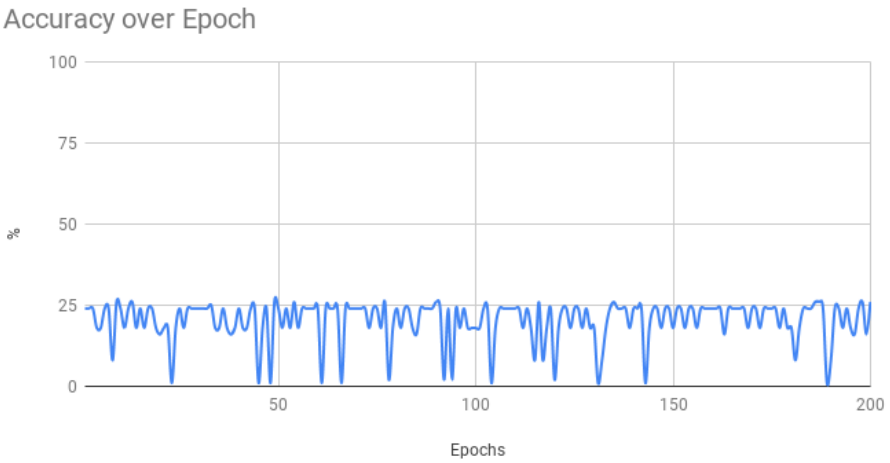

Figure 1.9: Accuracy over Epoch.

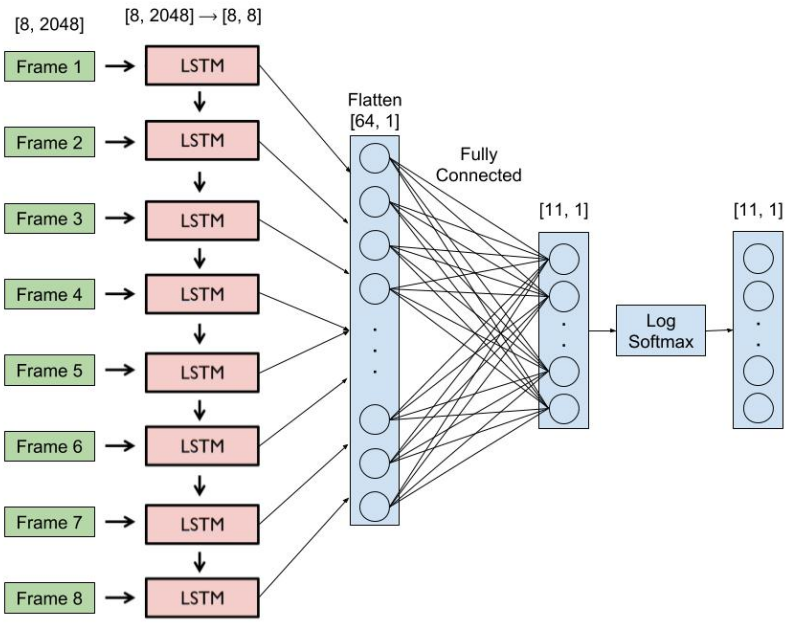

Figure 1.10: LSTM Classification v1.

Version 2

I had edited some parameter of my network architecture and change the the learning rate as well as the number of epochs. I changed the output size of Long - Short Term Memory cells from *[8, 8]* to ***[8, 1024]*** to determine whether I would have better results or not. I also decreased the learning rate to ***1e-5*** and the number of epochs to ***90*** (the number of epochs was actually 100 but I got my Colab session shut down and the loss was 0 anyway so I kept that number at 90).

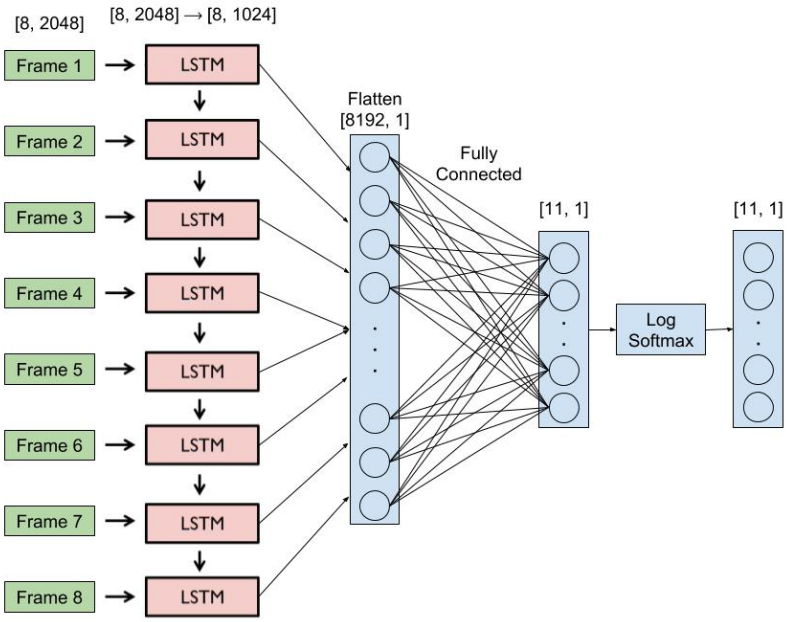

Figure 1.11: LSTM Classification v2.

I did not understand why after modifying the parameter of the network and learning rate, the initial loss value was drop from 89 to 1.8. The loss value was dropping over time as I expected, but unfortunately the accuracy was dropping too. At the time I got my loss value of 0, my test accuracy was just ***21.6%***. The maximum accuracy was also ***26%*** and the minimum value of loss was ***0*** but those values were not reliable. These figures below described the loss value and accuracy over epoch.

This architecture took me around ***2 hours*** to train 90 epochs (***1.3 mins*** per epoch).

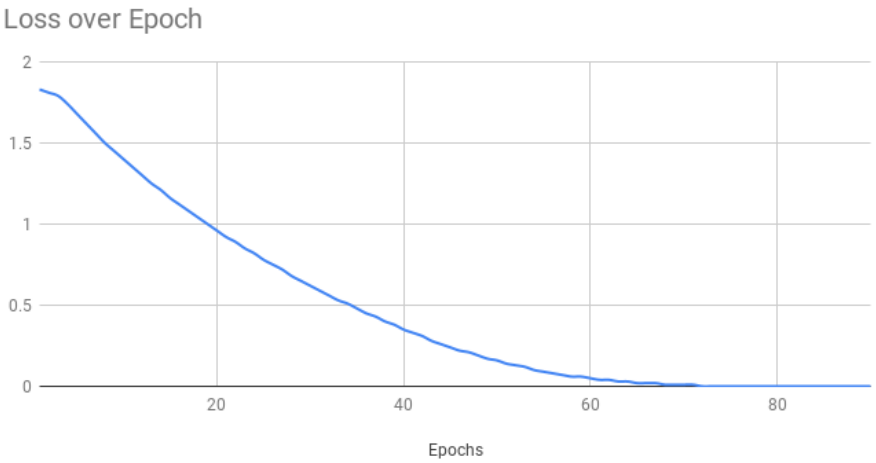

Figure 1.12: Loss over Epoch.

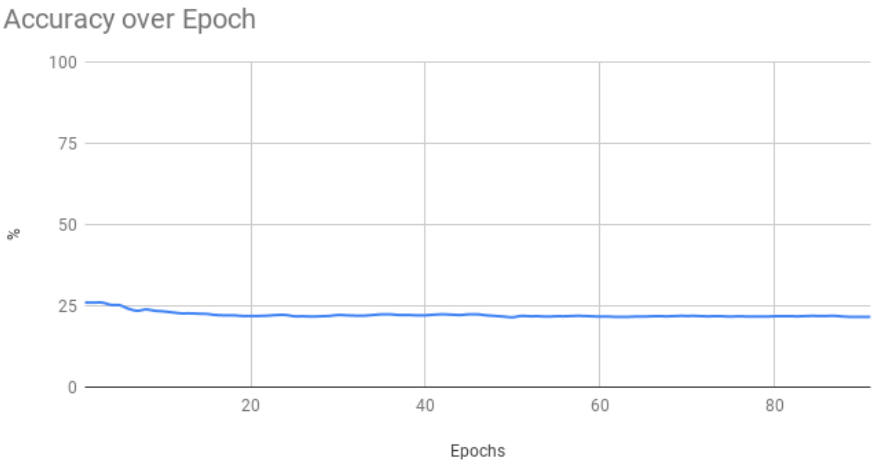

Figure 1.13: Accuracy over Epoch.

1.4.3 Training on LaMem-Set (extracted by Inceptionv3)

Introduction

Large - scale Image Memorability[61] was a dataset from MIT containing original 60000 images from diverse sources and corresponding memorability scores for memorability predicting purpose. The authors also introduced a Convolutional Neural Network with fine - tuned deep features that outperformed all other features by a large margin, reaching a rank correlation of 0.64, near human consistency (0.68). The authors could generate a memorability maps (likely Class Activation Map) for each image by using the analysis of the responses of the high - level CNN layers to shown which objects and regions were positively, and negatively, correlated with memorability.

Strategy

I used the same architecture as the Dev-set version v2 above to test this dataset. As the dataset had 55000 images when I downloaded it, I splitted this dataset into two parts, 45000 images for training and 10000 images for testing. I also have to change the number of Long - Short Term Memory cells in my network architecture to 1 because I had only one image at a time. I used Tensorflow's pre-trained Inceptionv3 convolutional neural network to convert those images above into feature vectors and then used them as the input of my network.

Version 1

I used the learning rate of ***1e-5*** and the number of epochs of ***20*** (it was a small number because this dataset was huge and took so much time to deal with, and my purpose was just to test my network architecture). This figure below demonstrated how my network architecture was.

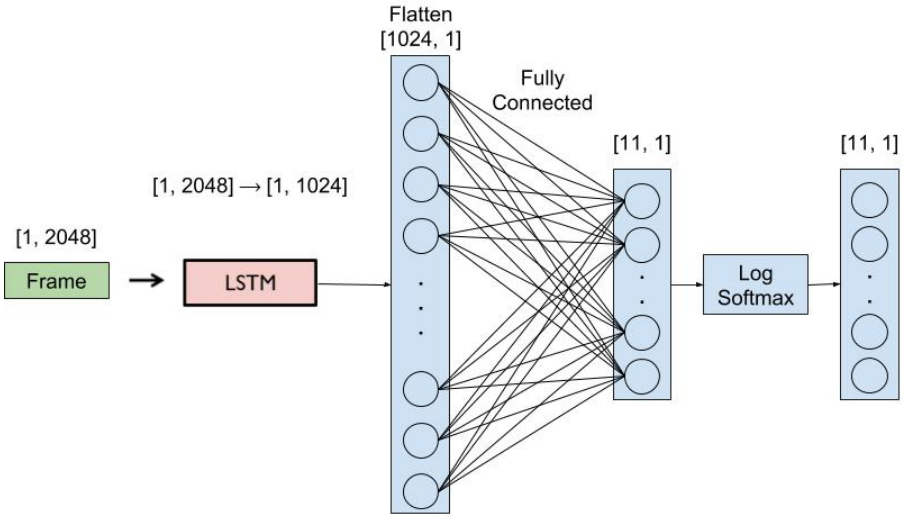

Figure 1.14: LSTM Classification for LaMem v1.

Because I only trained my network on this dataset for 20 epochs so the loss value was nearly remain from the initialization. In the other hand, the accuracy was too turbulent also. So I guess that my network architecture had some defects so it was not stable at all. The maximum accuracy was ***31%*** and the minimum value of loss was ***16.26***. These figures below described the loss valueand accuracy over epoch.

It took me around ***1.8 hours*** to train 20 epochs (***5.5 mins*** per epoch).

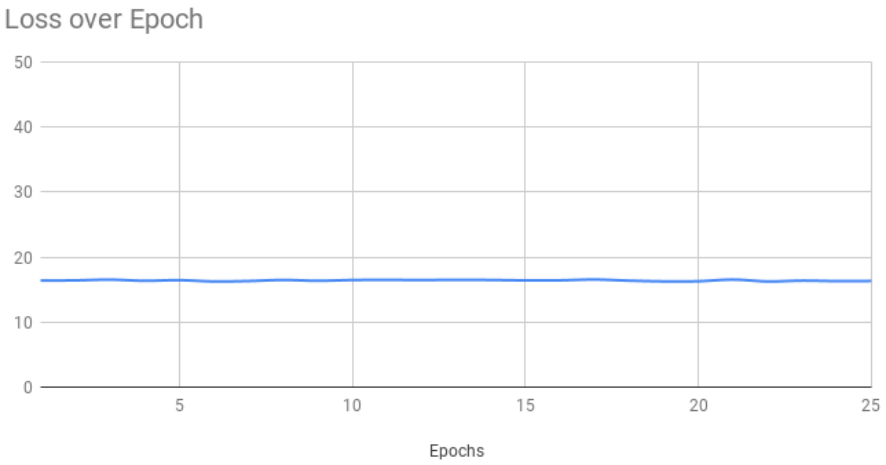

Figure 1.15: Loss over Epoch.

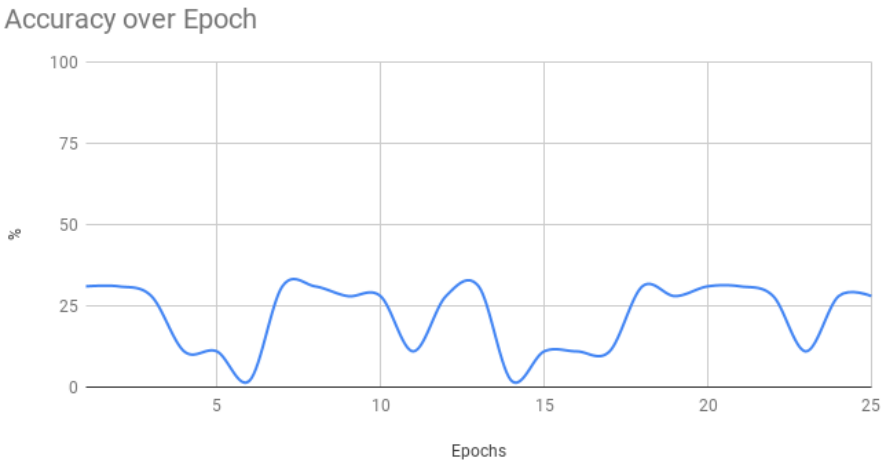

Figure 1.16: Accuracy over Epoch.

Chapter 2

Week 11 - August 16 2018

2.1 Introduction

I have spent the majority of my time this week trying my Long - Short Term Memory network with features extracted from Places365's pre-trained ResNet18, learning how to implement LSTM in TensorFlow.

2.2 Places365 Caption

This week I also run Places365 on images to analyze scene categories and attributes. And I concatenated 10 most relevant attributes structuring caption for a concrete image. Finally we tried to used that caption as input of any Convolutional Neural Network to predict images' memorabilities. I rewrote the source code and it could be found here on my GitLab.

2.3 TensorFlow LSTM Classification Network

2.3.1 Architecture

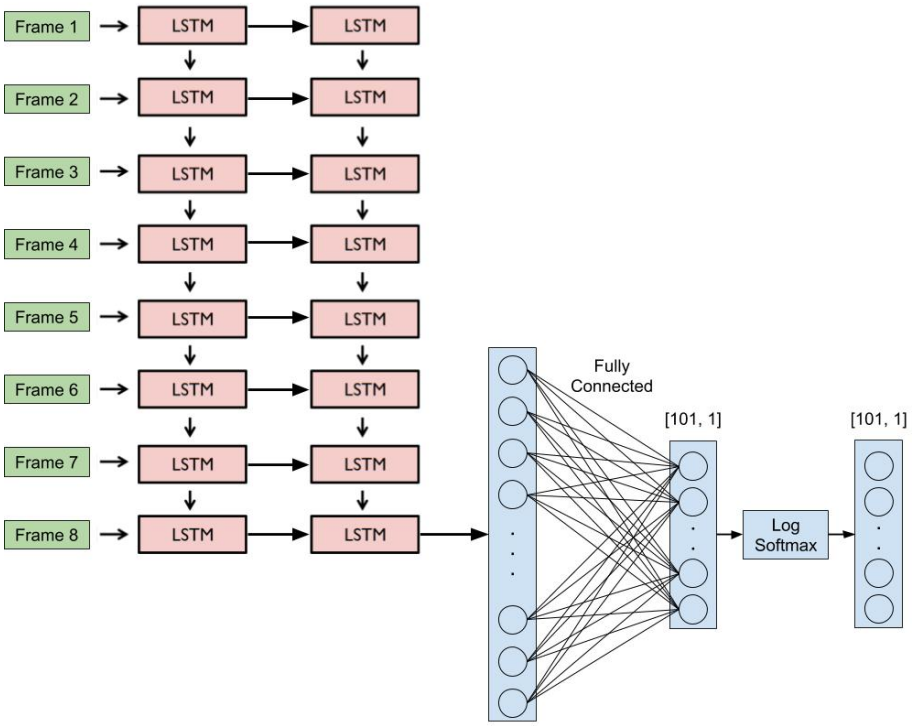

Figure 2.1: Long - Short Term Memory Network Architecture.

This week I tried to implement Long - Short Term Memory network using TensorFlow library. I finally figured out what was the definition of graph and session term in TensorFlow. I also learnt some new TensorFlow functionalities

for example Graph, Session, DropoutWrapper, BasicLSTMCell, MultiRNNCell, AdamOptimizer and argmax.

I re-designed my LSTM network using TensorFlow library and the figure below would demonstrate clearly how my network was. I used a stack of 2 LSTM cells sequences and only took the last LSTM cell's output for my Fully - connected layer. In each version I changed some hyper parameters to figure out whether I could increase the accuracy of not.

2.3.2 Training on Dev-Set (extracted by ResNet50)

Strategy

This times I broke the problem into the classification problem of 101 categories. Because I thought this dataset was pretty small for deeplearning so I splitted the provided dev-set for this challenge into two parts, since the dev-set had 8000 videos, I picked ***7000*** videos for training and ***1000*** for testing.

Version 1

In this version, I trained a batch of size ***128*** at once, with ***512*** LSTM units for each LSTM cell and ***7000*** iterations.

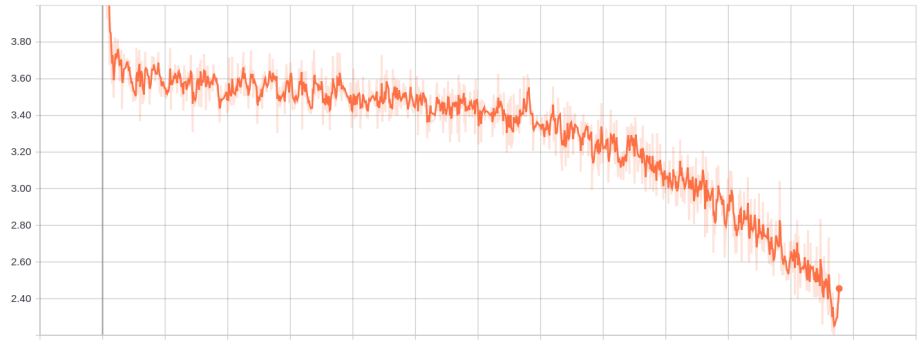

Figure 2.2: Loss over Epoch.

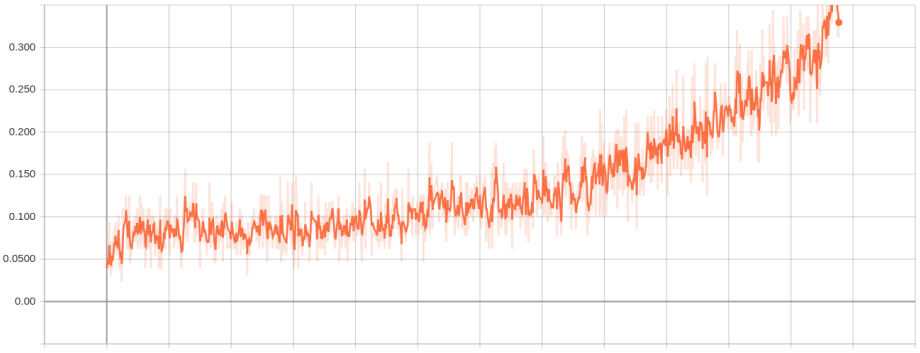

Figure 2.3: Accuracy over Epoch.

After 7000 iterations, the loss value was ***2.5*** and the accuracy value was ***0.3***. But from my perspective view, the loss and accuracy were both turbulent so lately I moved to another version with some hyper-parameter changed. After performing test on the ***train set***, I got the Spearman Rank Correlation of ***0.37***; on ***test set***, the Spearman Rank Correlation was just ***0.05***.

Version 1.1

In this revamp version, I trained a batch of size ***512*** at once, with ***1024*** LSTM units for each LSTM cell and ***7000*** iterations.

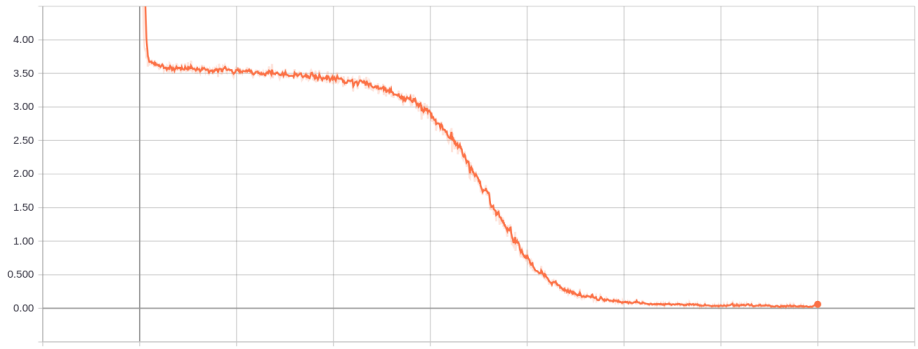

Figure 2.4: Loss over Epoch.

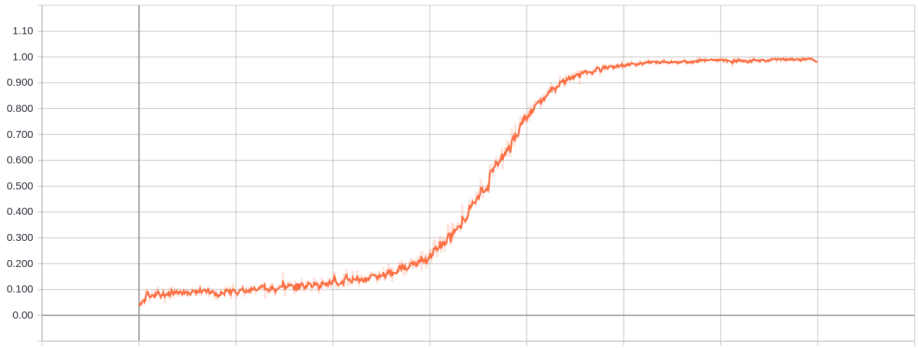

Figure 2.5: Accuracy over Epoch.

After 7000 iterations, the loss value was ***0.06*** and the accuracy value was ***0.98***. But from my perspective view, the loss and accuracy were both turbulent so lately I moved to another version with some hyper-parameter changed. After performing test on the ***train set***, I got the Spearman Rank Correlation of ***0.98***; on ***test set***, the Spearman Rank Correlation was just ***0.03***. This result seemed to be the consequence of overfitting. I thought 7000 iterations was not a good number.

2.3.3 Training on Dev-Set (extracted by Places365)

Strategy

I also broke the problem into the classification problem of 101 cate-gories. I first passed all of my input through Places365's pre-trained ResNet18 network and used those features as input of my LSTM network. The features extracted by ResNet18 had the dimension of ***512***. I aslo picked ***7000*** videos for training and ***1000*** for testing too.

Version 1

In this version, I trained a batch of size ***128*** at once, with ***256*** LSTM units for each LSTM cell and ***7000*** iterations.

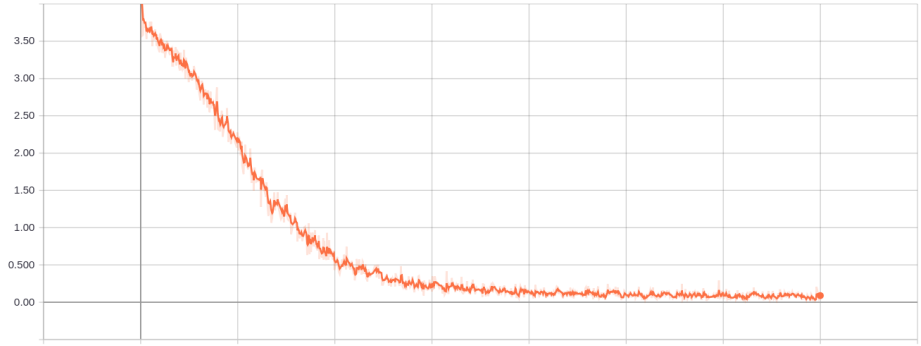

Figure 2.6: Loss over Epoch.

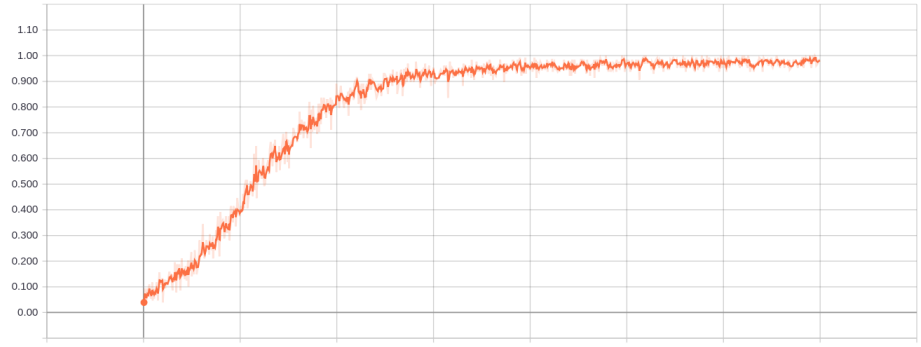

Figure 2.7: Accuracy over Epoch.

After 30000 iterations, the loss value was ***0.09*** and the accuracy value was ***0.98***. But from my perspective view, the loss and accuracy were both turbulent so lately I moved to another version with some hyper-parameter changed. After performing test on the ***train set***, I got the Spearman Rank Correlation of ***0.97***; on ***test set***, the Spearman Rank Correlation was just ***0.07***. This result also seemed to be the consequence of overfitting. I should decrease the number of iterations in the next version.

Bibliography

[1] École Polytechnique Fédérale de Lausanne, *EE-559 - Deep Learning.*

[2] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, A. Torralba, *Places: A 10 million Image Database for Scene Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[3] Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor, *Caffe: Convolutional Architecture for Fast Feature Embedding*, arXiv:1408.5093, 2014.

[4] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, *PyTorch.*

[5] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks.*

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, *Deep Residual Learning for Image Recognition*, arXiv:1512.03385, 2015.

[7] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, *Densely Connected Convolutional Networks*, arXiv:1608.06993, 2016.

[8] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba, *Learning Deep Features for Discriminative Localization*, arXiv:1512.04150, 2015.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, *Going Deeper with Convolutions*, arXiv:1409.4842, 2014.

[10] École Polytechnique Fédérale de Lausanne *EE-559 - Deep Learning*, *Mini-project 1: Prediction of finger movements from EEG recordings.*

[11] Karen Simonyan, Andrew Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, arXiv:1409.1556, 2014.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, *You Only Look Once: Unified, Real-Time Object Detection.*

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, arXiv:1311.2524, 2013.

[14] G. C. Benjamin Blankertz and K.-R. R. Muller, *Towards brain computer interfacing*, *Neural Information Processing Systems (NIPS)*, 2002.

[15] Davis E. King, *DLib*.

[16] Stanford University, CS231n - Convolutional Neural Networks for Visual Recognition.

[17] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *CIFAR-10 dataset*.

[18] Sergey Ioffe, Christian Szegedy *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*.

[19] Conference on Computer Vision and Pattern Recognition 2017.

[20] International Conference on Computer Vision 2017, *2017.thecvf.com*.

[21] Conference on Computer Vision and Pattern Recognition 2018.

[22] Eric Tzeng, Judy Hoffman, Kate Saenko, Trevor Darrell, *Adversarial Discriminative Domain Adaptation*, arXiv:1702.05464, 2017.

[23] Artem Rozantsev, Mathieu Salzmann, Pascal Fua, *Beyond Sharing Weights for Deep Domain Adaptation*, arXiv:1603.06432, 2016.

[24] Mehdi Mirza, Simon Osindero, *Conditional Generative Adversarial Nets*, arXiv:1411.1784 2014.

[25] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas, *Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks*, arXiv:1612.03242, 2016.

[26] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas, *StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks*, arXiv:1710.10916, 2017.

[27] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, *Generative Adversarial Networks* arXiv:1406.2661, 2014.

[28] Han Zhang, Ian Goodfellow, Dimitris Metaxas and Augustus Odena, *Self-Attention Generative Adversarial Networks*, arXiv:1805.08318, 2018.

[29] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida, *Spectral Normalization for Generative Adversarial Networks*, arXiv:1802.05957, 2018.

[30] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros, *Paired Image-to-image translation using Conditional GAN*, arXiv:1611.07004, 2016.

[31] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro, *High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs*, CVPR 2018.

[32] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros, *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*, ICCV 2017.

[33] Zhu, Jun-Yan and Zhang, Richard and Pathak, Deepak and Darrell, Trevor and Efros, Alexei A and Wang, Oliver and Shechtman, Eli, *Toward Multimodal Image-to-Image Translation*, arXiv:1711.11586, Nov 2017.

[34] Xiaolong Wang, Abhinav Shrivastava, Abhinav Gupta, *A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection*, CVPR, 2017.

[35] Ross Girshick, *Fast R-CNN*, arXiv:1504.08083, 2015.

[36] Jun-Yan Zhu, *pytorch-CycleGAN-and-pix2pix*.

[37] Cameron Smith, *neural-style-tf*.

[38] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, *Image Style Transfer Using Convolutional Neural Networks*.

[39] Manuel Ruder, Alexey Dosovitskiy, Thomas Broxn, *Artistic style transfer for videos*, arXiv:1604.08610, 2016.

[40] Leon A. Gatys, Matthias Bethge, Aaron Hertzmann, Eli Shechtman, *Preserving Color in Neural Artistic Style Transfer*, arXiv:1606.05897, 2016.

[41] The Starry Night by Vincent van Gogh, 1889.

[42] Multimedia Evaluation 2018, *mediaeval2018*.

[43] Medico: The 2018 Multimedia for Medicine Task.

[44] Joint Photographic Experts Group.

[45] The 2018 Emotional Impact of Movies Task, *emotionalimpact*.

[46] The 2018 Predicting Media Memorability Task, *memorability*.

[47] James L McGaugh, *Memory-a Century of Consolidation*, Science 287, 5451 (2000), 248-251.

[48] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, *Learning Spatiotemporal Features with 3D Convolutional Networks*, ICCV 2015.

[49] Khurram Soomro, Amir Roshan Zamir and Mubarak Shah, *UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild*, CRCV-TR-12-01, November 2012.

[50] ALMEIDA, Jurandy; LEITE, Neucimar J.; TORRES, Ricardo da S, *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 2011. p. 3673-3676.

[51] Dalal, N. and B. Triggs, *Histograms of Oriented Gradients for Human Detection*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 1 (June 2005), pp. 886–893.

[52] DC. He and L. Wang (1990), *Texture Unit, Texture Spectrum, And Texture Analysis*, Geoscience and Remote Sensing, IEEE Transactions on, vol. 28, pp. 509 - 512.

[53] SZEGEDY, Christian et al, *Rethinking the inception architecture for computer vision*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. p. 2818-2826.

[54] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski, *Orb: an efficient alternative to sift or surf*, In Computer Vision (ICCV), 2011, IEEE International Conference on, 2011.

[55] OpenCV *Keypoints and Descriptors tutorial*.

[56] Eugenio Culurciello, TowardsDataScience, *Neural Network Architectures*.

[57] Sepp Hochreiter; Jürgen Schmidhuber, *Long Short-Term Memory*, Neural Computation.

[58] Google's AI organization, *TensorFlow*.

[59] Ronan Collobert, Koray Kavukcuoglu, Clement Farabet, *Torch*.

[60] Skymind, *From word to embeddings*.

[61] Aditya Khosla, Akhil S. Raju, Antonio Torralba and Aude Oliva, *Understanding and Predicting Image Memorability at a Large Scale*, International Conference on Computer Vision, 2015.

[62] Matt Harvey, Coastline Automation, *Five video classification methods implemented in Keras and TensorFlow*.

[63] Phillip Isola, Jianxiong Xiao, Antonio Torralba and Aude Oliva, *What makes an image memorable?*, IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[64] T. Konkle, T. F. Brady, G. A. Alvarez, and A. Oliva, *Conceptual distinctiveness supports detailed visual long-term memory for realworld objects*, JEP:G, 2010.

[65] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, *LabelMe: a database and web-based tool for image annotation*, IJCV, 2008.