#### PROJECT MARCH -APRIL 2021

A REPORT

ON

**Video Compression**

BY

|  |  |
| --- | --- |
| Bir Anmol Singh | 2018A7PS0261P |
|  |  |

On Research Paper

**Learning Binary Residual Representations for Domain-specific Video Streaming**

Prepared for Multimedia Computing

Course No.

CS F401

**ABSTRACT**

Today many online video streaming platforms have emerged. The video resolution has increased but the bandwidth growth has been significantly lower. Thus, all streaming services use **H.264 compression** and reduce **bit rate** to stream video smoothly. Many a times the compression is unable to detect important features from raw video and thus gives a blurry effect if the compression ratio is high and if the ratio is low, it can cause buffering.

Majority of the stream we do, belongs to one kind of domain – **domain specificity**. If we target such a streaming setting where the videos to be streamed from a server to a client are all in the same domain and they have to be compressed to a small size for low-latency transmission, one can leverage this property of domain specificity to achieve better video quality over the conventional compression. A new compression pipeline can be used where we first apply H.264 to compress domain-specific videos, then train a novel **binary autoencoder** to encode the leftover domain-specific **residual information** frame-by-frame into binary representations. These binary representations can be encoded in a **lossless manner by using Huffman encoding** and can be sent along with the H.264 stream to the user.

This can help us gain quality over the standard compressed video at similar/lower bandwidth.

**TABLE OF CONTENTS**

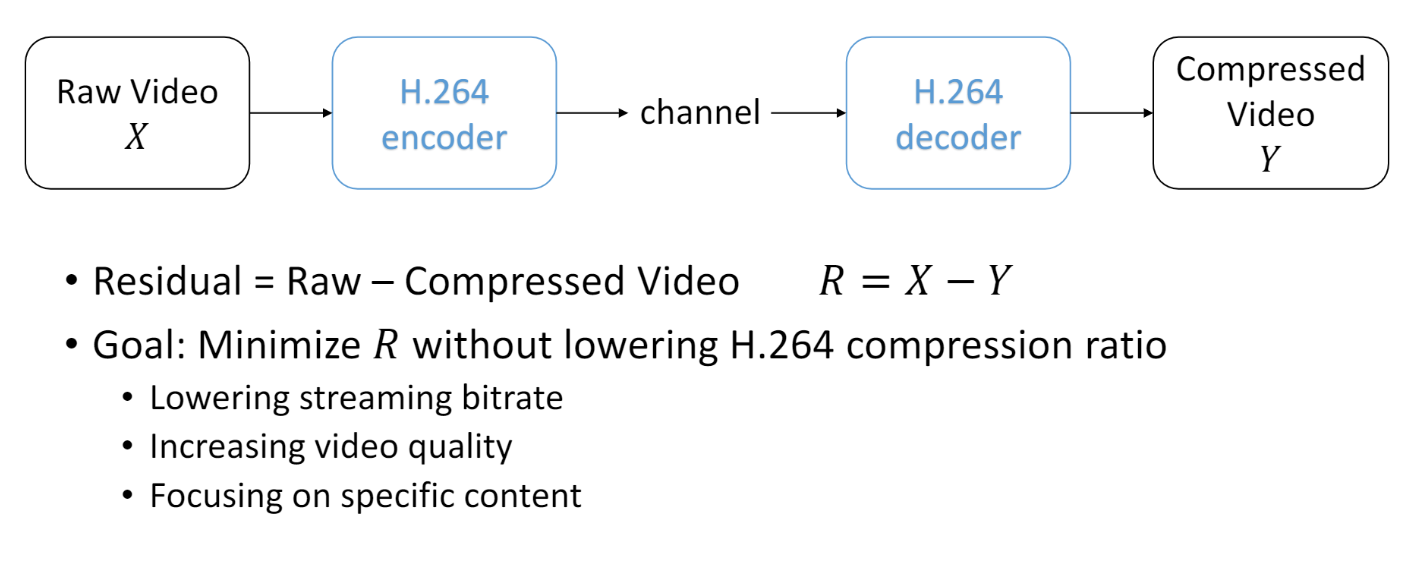
|  |  |
| --- | --- |
|  | Page No. |
| 1. Cover Page……………………………………………………………… 2. Abstract ………………………………………………………............... 3. Introduction ……..………………………………….............................. 4. Approach to Solution …………………………………………………. 5. Design of Solution …..………………………………………………... 6. Summary and Conclusion…………………............................................ 7. References …………………………………………………………....... | 1  2  4  7  8  11  12 |

**INTRODUCTION**

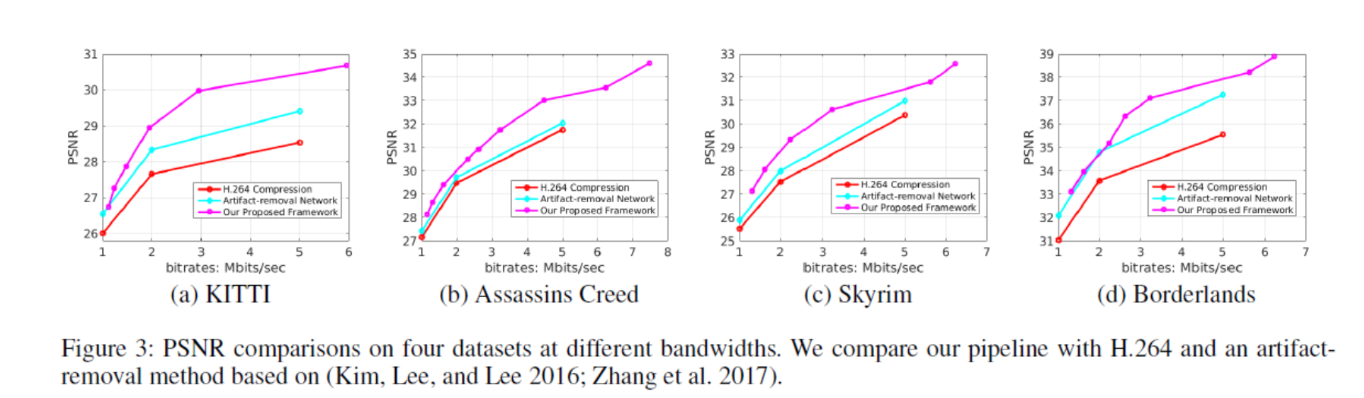
**About the Problem**

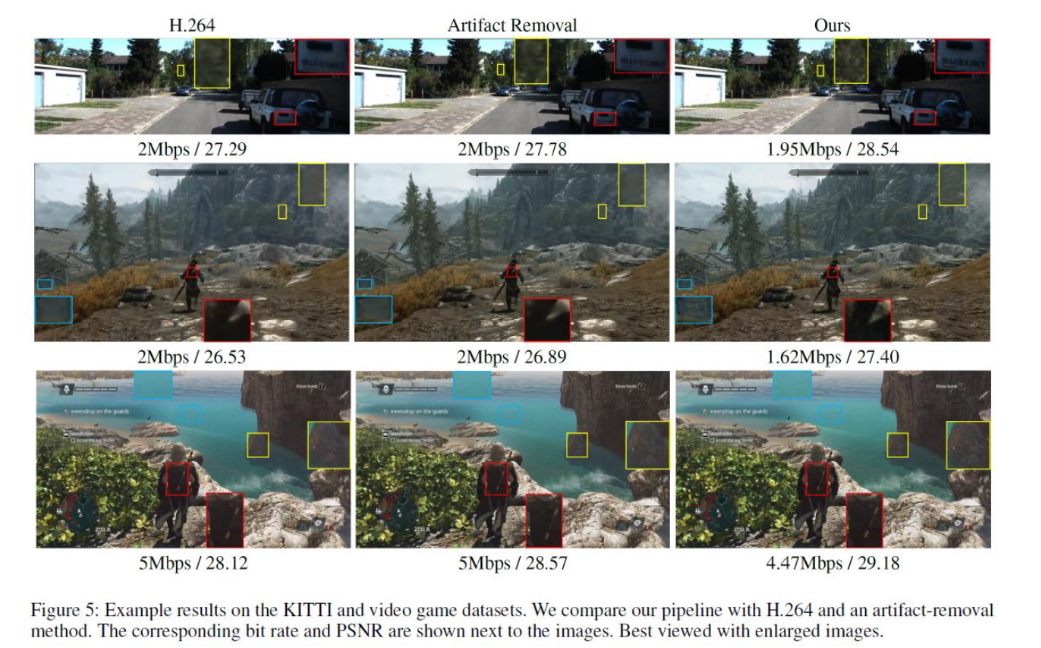
Video streaming services, such as Netflix and YouTube, have become primary source of entertainment nowadays. These websites have high resolution, large size videos which have to be sent over limited network bandwidth thus **video compression** is a necessary evil. While video compression can reduce the size of a video, it often comes with undesired compression artifacts, such as image blocking effects and blurry effects.

Video compressing has improved after decades of efforts on delivering the best possible video quality under bandwidth constraint. State of-the-art video compression methods such as **MPEG-4 (Li2001**), **H.264 (Wiegand et al. 2003**), and **HEVC (Sullivan et al. 2012)** combine various classical techniques including **image transform coding**, **predictive coding**, **source coding**, and **motion estimation** in a carefully-engineered framework. These methods are generic and can be applied to various video domains for effectively compressing most of the information in a video however, **the residual information**, which is the difference between the uncompressed and compressed videos, is difficult to compress because it contains highly non-linear patterns. This residual information can be preserved if compression is applied on stream belonging to a **specific domain**. Although this may appear as a lot of work since one needs to have a video compressor for each domain, but it can be used in cases like video game streaming and sports streaming. The content that is transmitted during stream time stays to one genre and thus can be exploited for residual compression. This residual information can be compressed and sent along with regular H.264 packets to get better quality at similar bandwidth levels.



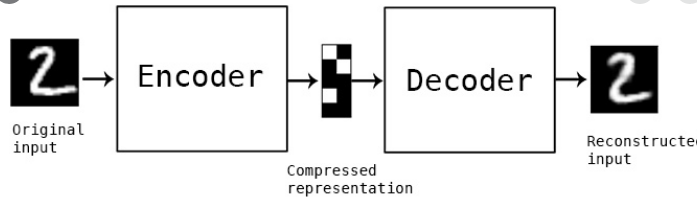
The research paper proposes expected results of same **Peak Signal to Noise Ratio (PSNR)** at lower bandwidth rate with various data sets.



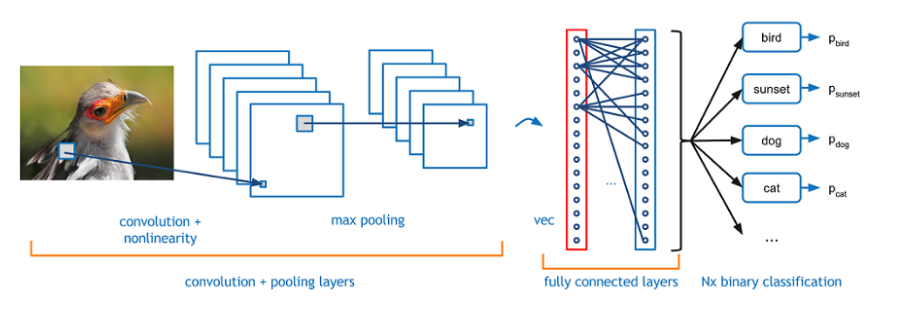


**Background Study**

The residual information present in video after compression is non-linear in nature. This non-linear information can be preserved by using predictors using **deep neural networks**. The feature extraction can be done by the **Convolutional neural networks (CNNs)**. Using **auto-encoder** i.e. an unsupervised artificial neural network that learns to compress and encode data and also learns how to reconstruct the data back from the encoded representation to an approximate original format (lossy). Adding this auto-encoder alongside with a H.264 compressed video to generate a larger residual signal and generating small encoding of this residual signal as bit stream to be sent along with H.264 packet as metadata. Auto encoders are more efficient at preserving the residual information but very slow and inaccurate for entire video, thus the need for generating this hybrid of H.264 and auto encoder that can fit well into existing framework.

Auto Encoder

Convolutional neural network



**APPROACHING THE SOLUTION**

We will leverage deep learning models for preserving/predicting the residual information of domain specific content. Neural Networks are powerful non-linear function approximators, which can be used to encode the highly nonlinear residual information.

In the video compression pipeline:

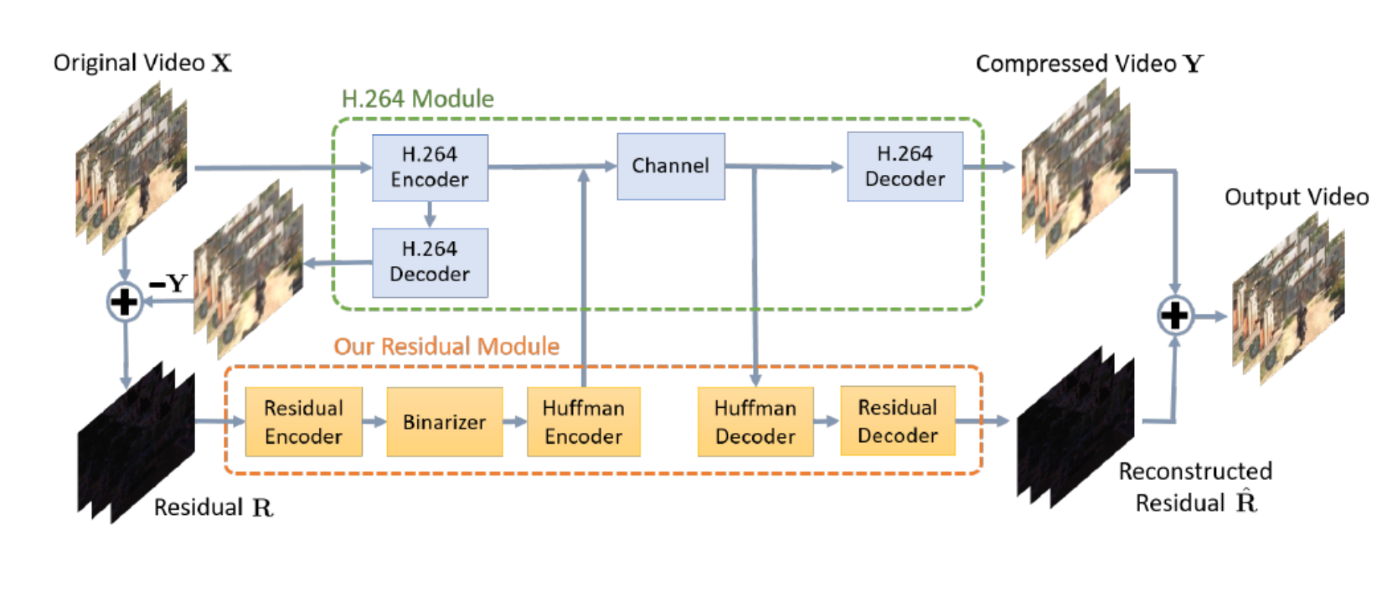
1. Apply H.264 to compress videos in a specific domain and train a novel binary autoencoder to encode the resulting residual information frame-by-frame into a binary representation.

2. Apply Huffman coding to encode the binary representations in a lossless manner.

3. The compressed binary representations are sent to the client in the meta data field in the H.264 streaming packet.

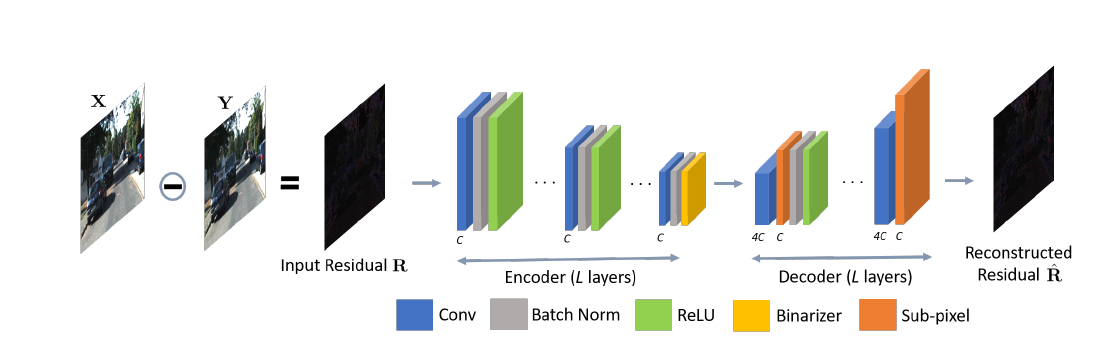
4. The decoding of this data can be done and video can be regenerated.

For example: we have 5MB/s link, we can use high compression ratio-based H.264 to get noisy video at 4MB/s and remaining 1MB/s can be taken up by our Huffman encoded meta data. The auto-encoder generates smaller size byte stream for residual information, which is also significant due to higher compression ratio of H.264. Thus we can send more information in similar bandwidth levels and club it up to generate better looking video at client side.



**DESIGN FOR THE SOLUTION**

Design for Auto-encoder:



Layers of CNN = 3 (L)

Activation applied: ReLu

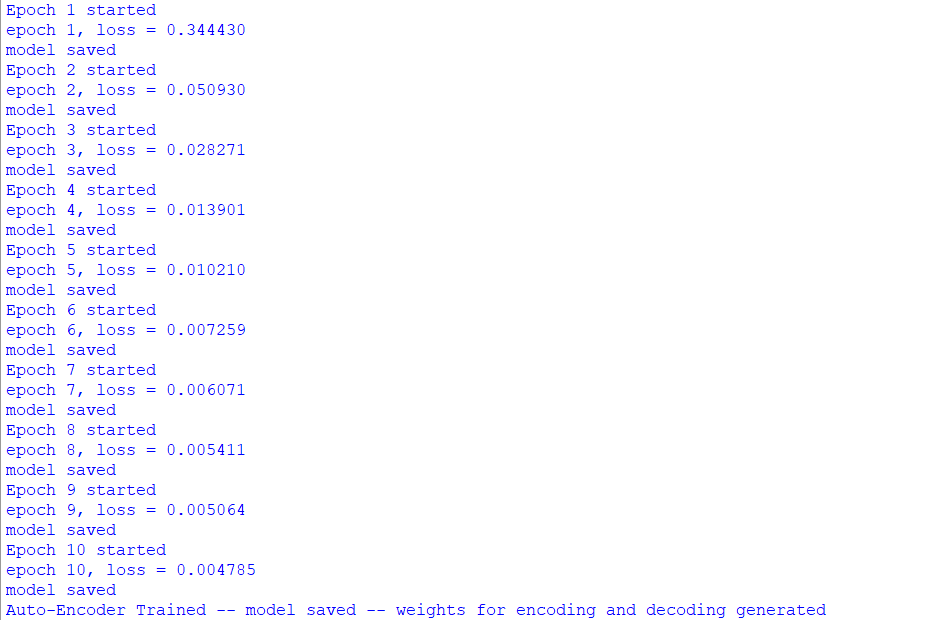
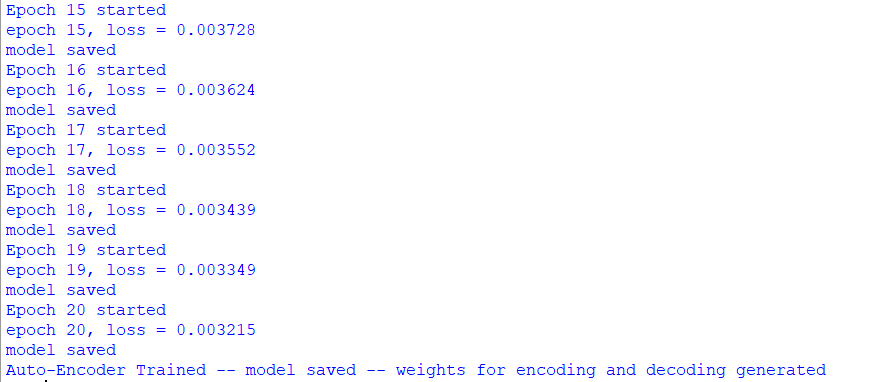
Loss optimizer used: Adam

Training done on Football\_raw.mp4 at different compression ratio to get different Football\_compressed.mp4 files [times- 6 sets]

Epochs = 10 per training

Batch Size =10 , BITS per Group =16

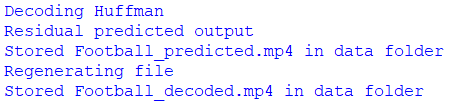
The encoder and decoder are derived from auto-encoder itself.

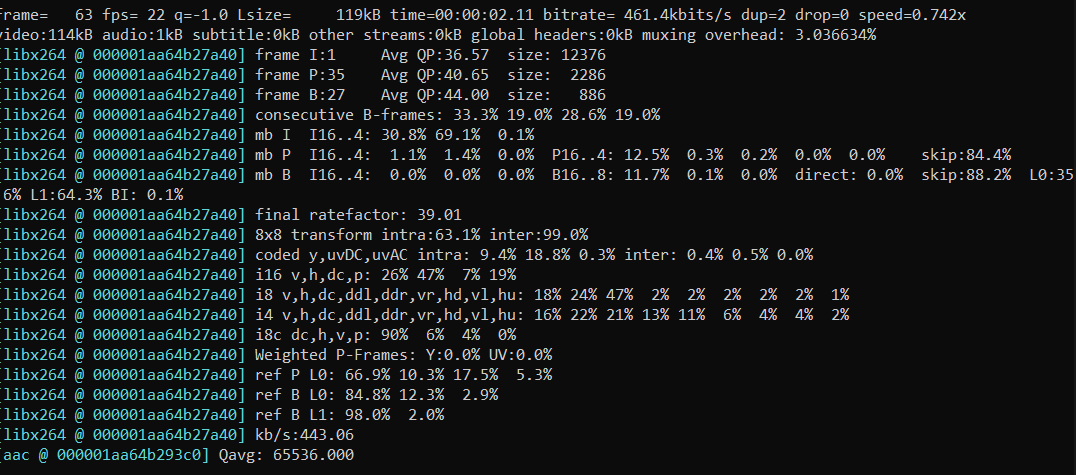
 

Implemented Huffman encoding to encode residual information in a lossless manner



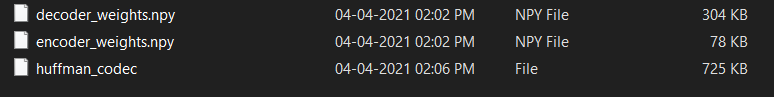
Decoding this information to obtain the final results



Using ffmpeg to control final bit-rate 

**Results:**

The H.264 compressed video generated from RAW video is of 234KB.

The Huffman encoding generated metadata for residual values is of 725KB

Effective data: 725+234 = 959KB whereas raw bandwidth we could use was 1205KB.

We can decode and adjust bit rate using FFmpeg to obtain ~900KB result.

Residual video shows the motion predicted

 **RAW Image**

 **H.264 Image**

At 350x zoom level H.264 Football contour at left end is missing whereas our algorithm preserves this information

 **Our Image**

**SUMMARY and CONCLUSION**

The visual differences obtained are significant if the stream is really detailed and fast-moving objects. The artifaction and blurring can be mitigated by having a good residual prediction by using a CNN with multiple layers trained on big data sets. But the caveat here is that this needs specialized hardware and is only applicable for niche domains.

The impracticality of this approach can be seen from my testing. 10 epoch training on a 2sec 1920\*1080 (1080p) video takes 1hr (even after turning on specialized checkpointing). Thus, for higher resolution videos and longer lengths, the training can take several days.

The bits per group used to encode meta data brings better compression ratio but also affects the overall encoding size. Higher number of bits yielded costly size rise of residual encodings.

The encoding and decoding are significantly slow and thus not useful for real time streams.

However, if there is use of specialized GPUs and pretrained models are well built, the technique can very easily be added to the existing framework making it lucrative for pre-record and present type of streams. With power of parallel processing and multi-core CPUs, the decoding and encoding can be made faster.

**Other problems I faced:**

Lack of datasets for proper training and weak GPU thus reduced training and low quality residual generation.

**REFERENCES**

**Research reference:**

* Tsai, {Yi Hsuan} and Liu, {Ming Yu} and Deqing Sun and Yang, {Ming Hsuan} and Jan Kautz; Learning binary residual representations for domain-specific video streaming ; AAAI press.; 32nd AAAI Conference on Artificial Intelligence, AAAI 2018 ; 2018; 7363-7370

[link: <https://research.nvidia.com/sites/default/files/pubs/2018-02_Learning-Binary-Residual/Learning%20Binary%20Residual%20Representations%20for%20Domain-specific%20Video%20Streaming.pdf> ]

**Other References:**

* **NVIDIA YouTube video**: <https://www.youtube.com/watch?v=SCc0SqzkQf4>
* **Auto Encoders**: [Auto-Encoder: What Is It? And What Is It Used For? (Part 1) | by Will Badr | Towards Data Science](https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726)
* **CNNS**: [Understanding CNN (Convolutional Neural Network) | by Vincent Tatan | Towards Data Science](https://towardsdatascience.com/understanding-cnn-convolutional-neural-network-69fd626ee7d4)
* <https://github.com/aeric1102/Residual-Encoding-for-Domain-Specific-Video>
* [GitHub - atriumlts/subpixel: subpixel: A subpixel convnet for super resolution with Tensorflow](https://github.com/atriumlts/subpixel)