Read the dissertation online! https://maxehr.umiacs.io/dissertation

The First Principles of Deep Learning and Compression

Dissertation by Max Ehrlich

Chair Professor Abhinav Shrivastava, Advisor
Co-Chair Professor Larry S. Davis, Advisor
Dean's Representative Professor Wojciech Czaja
Examining Committee
Professor Ramani Duraiswami
Professor Dinesh Manocha
Dr. Michael A. Isnardi (SRI International)
Professor David A. Forsyth (UIUC)

maxehr@umiacs.umd.edu
https://maxehr.umiacs.io
https://scholar.google.com/citations?user=q-WSy3AAAAAJ

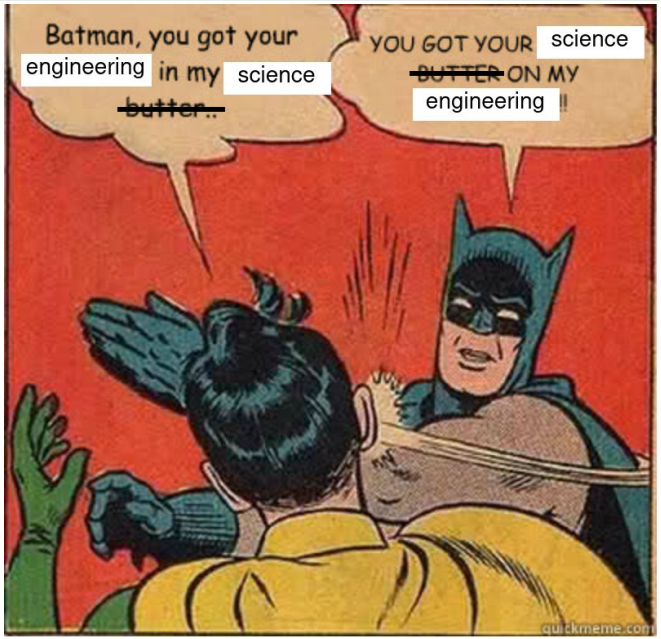
UNIVERSITY OF MARYLAND

In the history of AI/pattern recognition, ***ad-hoc*** *engineered methods have often worked better initially, before enough data and compute power was available to rely increasingly on ML. It happened in OCR, ASR, CV, MT, NLP, and now in robotics*

- Yann LeCun (Twitter, ca. 2021)

Engineering vs. Science?

- "Traditional" pipeline:
  - Scientists do the science
  - Engineers make something work from the science
- Science informs engineering, can engineering inform science as well?
- Synergistic feedback loop
  - Science → Engineering → Better Science → etc.
- Many interesting scientific problems are grounded in engineering
  - Today we'll talk about compression, but there are many more
- **General ML based methods are important and interesting!**
  - But they're not what I study and that's ok!

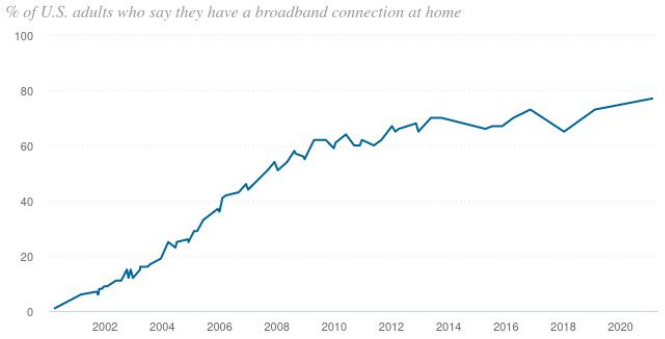


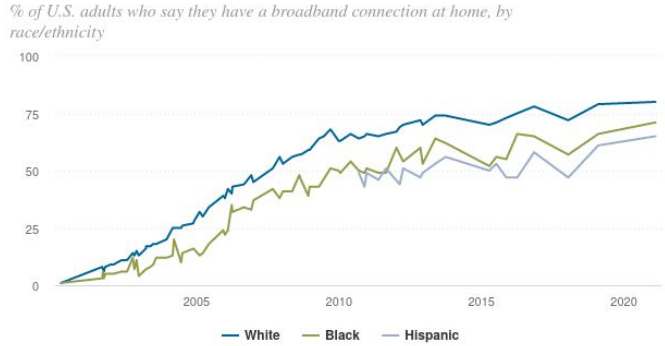
The First Principles Approach to Science

- When working with an engineering motivated problem
  - Like compression related problems
- There were myriad engineering decisions that influence the design
  - We call these decisions the **“First Principles”** of the problem
- We take these first principles into account when developing new algorithms
  - These algorithms are customized to the specific problem
  - But what we lose in generality we make up for in results

The Internet

- The internet is unprecedented
  - Entertainment medium
  - Repository of human knowledge
- Approximately a quarter of Americans do not have a broadband connection
  - Hits rural, less educated, and minorities the worst
  - Many rural areas use metered connections
  - Problem not restricted to America
- Society as a whole benefits from maximal participation in the internet

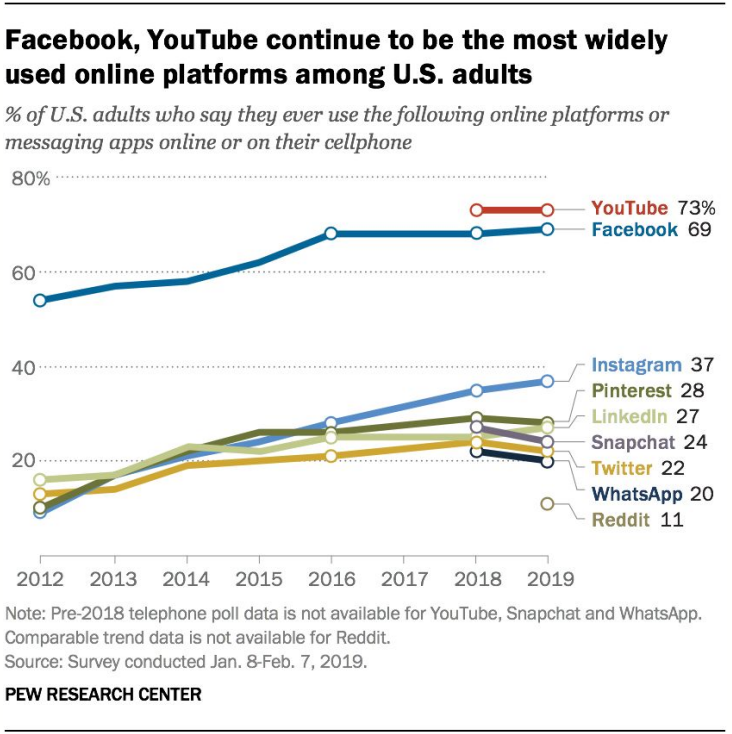




Graph credit: Pew Research

Multimedia

- Internet has coalesced around images and videos
  - Most popular sites are centered around image/video sharing
  - Huge growth since 2012
- Video is growing particularly fast
  - 78% of people watch videos online every week, 55% every day
  - Projected: 82% of consumer internet traffic will come from videos by 2022



Source: https://breadnbeyond.com/video-marketing/video-marketing-strategies-statistics/

Multimedia Compression

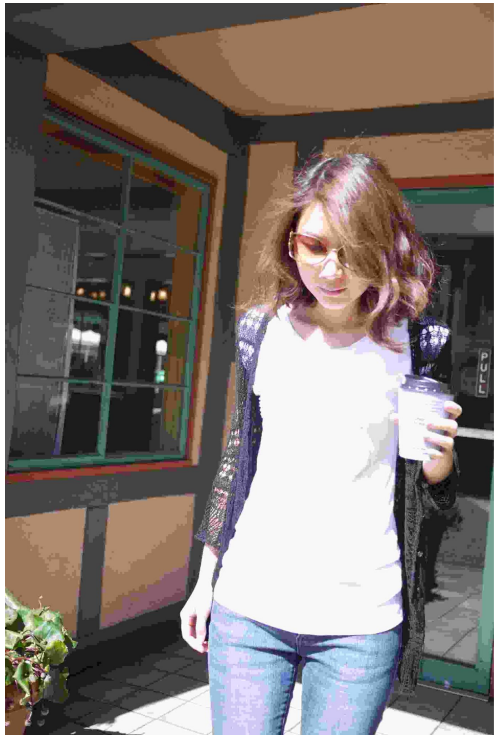
- Lack of broadband often precludes sharing and viewing of multimedia, something which is increasingly prerequisite to modern internet use
  - Metered connections make this even harder
- Almost all multimedia are compressed to help with this
  - JPEG (among others) for images
  - MPEG/AOM for videos (most common is H.264)
- H.264 covers around 91% of all videos as of 2019 [1]
  - Was first released in 2003 (19 years old)
  - H265, AV1, etc do better but aren't common
- Global Pandemic
  - School and work are online!

1. "Video Developer Report 2019". Bitmovin. September 2019.

Deep Learning and Multimedia Compression

- Deep learning has revolutionized computer vision
  - Compression is no exception
- Example: “HiFiC” [1]
- So why aren’t we using this?
  - Video and image codecs are entrenched in complex pipelines
  - Codec development is slow and deliberative

HiFiC: 0.082bpp



JPEG: 0.087bpp

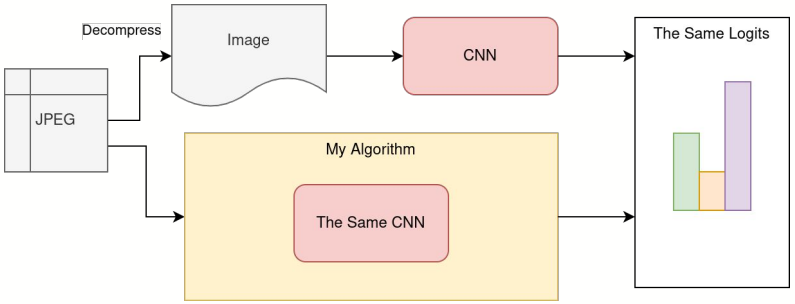
1. Mentzer, Fabian, et al. "High-Fidelity Generative Image Compression." *NeurIPS*. 2020.

My Dissertation

- Instead of trying to replace classical codecs with deep learning, let's work *with* them
- Big advantage: can be used in real applications in the near-term
  - Only depending on classical compression means no special software
- So many great problems to solve:
  - Compressed domain DL
  - Improve classical compression
  - Improve DL accuracy on compressed images
- In all cases: leverage what we **know** about compression to inform our science
  - The engineers that developed these compression standards are smart, **their decisions** should guide **our decisions**

Deep Residual Learning in the JPEG Transform Domain

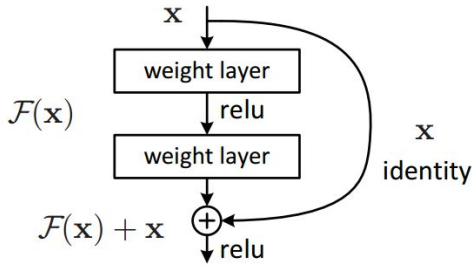
- JPEG decompression takes time, and JPEG data is sparse
- If convolutional networks could work on compressed JPEG data
  - We could leverage the sparsity
  - And avoid decompressing the image



- Goal of this work: reformulate residual networks such that they work on compressed JPEG data
  - Should be as close as possible to the spatial domain network
- Need to merge two mathematically disjoint theories
  - Deep learning with convolutions
  - JPEG compression, specifically the DCT

Solution - First Principles

- Our formulation leverages the linearity of the JPEG transform
  - The linear parts of convolutional networks can either be composed with the JPEG transform to create a new linear map or reformulated directly
- Simplified ResNet
  - Some ResBlocks
  - Global average pooling
  - Fully connected layers
- Need to reformulate the following operations
  - Convolution - linear
  - Batch Norm - statistics of JPEG transform
  - Sum - linear
  - Global average pooling - statistics of JPEG transform
  - ReLU - Uh Oh!
- ReLU is piecewise linear, so we can't compose it
  - So we use a novel approximation technique



Quick Overview of JPEG

Compression

1. Compute the DCT of non-overlapping 8 x 8 blocks
2. Divide the DCT coefficients by a quantization matrix and round them to integers
3. Vectorize the quantized coefficients putting high frequencies at the end
4. Run-length code and entropy code

Decompression

1. Undo entropy coding, run-length coding
2. Rearrange vectors into 8 x 8 blocks
3. Multiply coefficients by the quantization matrix
4. Inverse DCT

Making it Linear: Definition of JPEG Transform Domain

Compression

1. Compute the DCT of non-overlapping 8 x 8 blocks
2. Divide the DCT coefficients by a quantization matrix and ~~round them to integers~~
3. Vectorize the quantized coefficients putting high frequencies at the end
4. Run-length code and entropy code

Decompression

1. Undo entropy coding, run-length coding
2. Rearrange vectors into 8 x 8 blocks
3. Multiply coefficients by the quantization matrix
4. Inverse DCT

JPEG Compression Tensors

Method of Brian Smith (1994) [1], can compute any linear function in JPEG domain

- Multi-channel batch of images/features: type (0, 4) Tensor

$$I \in B^* \otimes C^* \otimes H^* \otimes W^*$$

- Transform Coefficients

$$\Phi \in B^* \otimes C^* \otimes X^* \otimes Y^* \otimes K^*$$

Blockifying: $$B^{hw}_{xymn} = \begin{cases} 1 & h, w \text{ belongs in block } x, y \text{ at offset } m, n \\ 0 & \text{otherwise} \end{cases}$$

DCT: $$D_{\alpha\beta} = \frac{1}{4} C(\alpha) C(\beta) \cos\left[\frac{(2m+1)\alpha\pi}{16}\right] \cos\left[\frac{(2n+1)\beta\pi}{16}\right]$$

Zigzag: $$Z^{\alpha\beta}_{\gamma} = \begin{cases} 1 & \alpha, \beta \text{ is at } \gamma \text{under zigzag ordering} \\ 0 & \text{otherwise} \end{cases}$$

Quantization: $S^{\gamma}_{k} = \frac{1}{q_k}$ Dequantization: $\widetilde{S}^{k}_{\gamma} = q_k$

JPEG Compression: $$J^{hw}_{xyk} = B^{hw}_{xymn} D^{mn}_{\alpha\beta} Z^{\alpha\beta}_{\gamma} S^{\gamma}_{k}$$

JPEG Decompression: $$\widetilde{J}^{xyk}_{hw} = B^{xymn}_{hw} D^{\alpha\beta}_{mn} Z^{\gamma}_{\alpha\beta} \widetilde{S}^{k}_{\gamma}$$

1. Smith, B. "Fast software processing of motion JPEG video." Proceedings of the second ACM international conference on Multimedia. 1994.

Convolutions

NLS Method:

Naive Steps:

1. Decompress
2. Convolve
3. Compress

Linear Maps:

1. $I_{hw} = \tilde{J}^{xyk}_{hw} \Phi_{xyk}$
2. $I'_{h'w'} = C^{hw}_{h'w'} I_{hw}$
3. $\Phi'_{x'y'k'} = J^{h'w'}_{x'y'k'} I'_{h'w'}$

Substitute:

$$\Phi'_{x'y'k'} = \left[J^{h'w'}_{x'y'k'} C^{hw}_{h'w'} \tilde{J}^{xyk}_{hw} \right] \Phi_{xyk}$$

Let:

$$\Xi^{xyk}_{x'y'k'} = J^{h'w'}_{x'y'k'} C^{hw}_{h'w'} \tilde{J}^{xyk}_{hw}$$

Then:

$$\Phi'_{x'y'k'} = \Xi^{xyk}_{x'y'k'} \Phi_{xyk}$$

- $\Xi^{xyk}_{x'y'k'}$ is the compressed domain convolution
- This tensor encapsulates decompression, applying the convolution and recompression
- Easily extended to multi-channel convolutions

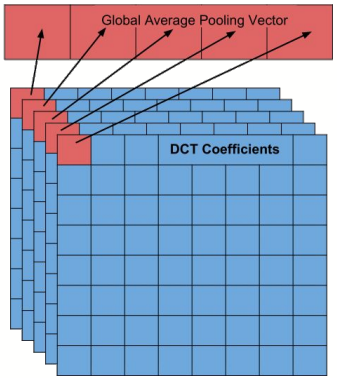
Batch Norm and Global Average Pooling

Batch Norm

- Defined as $\mathrm{BN}(I) = \gamma \frac{I - \mathrm{E}[I]}{\sqrt{\mathrm{Var}[i]}} + \beta$ for learnable gamma and beta
- We can use two properties of the DCT:
  - the (0, 0) coefficient in each block is the mean of the block and the expectation of the coefficients is the variance of the block.
  - This lets us extract the mean with a simple read operation and apply beta with a single addition
- To apply gamma and the variance, we use the linearity of JPEG

Global Average Pooling

- Represents each channel by the spatial mean
- As stated on the right, the mean is the (0,0) coefficient, so we simply read this off instead of computing something



ReLU

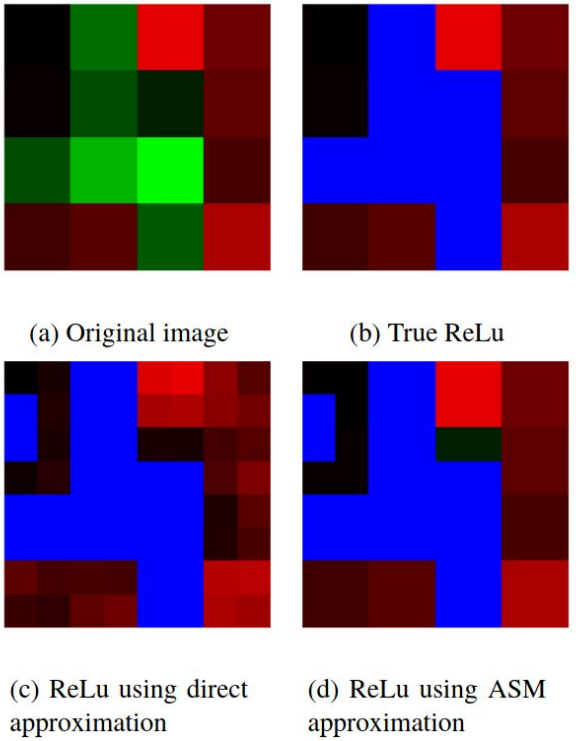
- This is a big problem because it's non-linear
  - So we can't easily compose it with the JPEG linear maps
- Sometimes you can do fancy algebra to make non-linear functions work
  - But ReLU is a worst case since it is piecewise linear, you need to know the spatial domain value to apply the right piece
- Idea: partially decompress the block
- If we want to approximate a signal using a subset of the DCT coefficients, the best coefficients to use are the lowest frequency M coefficients
  - We can use this to implement a fast approximate ReLU by doing a partial decompress
  - Decompressing only a few frequencies does a good enough job without sacrificing performance

$$R(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

$$R(x) = H(x)x$$

Approximated Spatial Masking

- One potential problem
  - Although the lowest M frequencies approximate the original signal, all of the signal values could still be wrong
- ReLU zeros out negative pixels, it would be nice if the positive pixels were at least preserved
- So instead of using the ReLU of the approximation as our result, compute a mask from the approximation and then apply the mask to the untouched DCT coefficients
- New problem
  - We partially decompress to compute the mask, so the mask is in the pixel domain
  - But we need to multiply it by DCT coefficients or we'd have to decompress the block anyway!



(a) Original image

(b) True ReLu

(c) ReLu using direct approximation

(d) ReLu using ASM approximation

Applying a Pixel Mask to DCT Coefficients

NLS Method

Naive Steps:

1. Inverse DCT
2. Pixelwise multiply
3. DCT

Linear (or Bilinear) Maps

1. $I_{mn} = D^{\alpha\beta}_{mn} F_{\alpha\beta}$
2. $I'_{mn} = I_{mn} G_{mn}$
3. $F'_{\alpha'\beta'} = D^{mn}_{\alpha'\beta'} I'_{mn}$

Substitute

$$F'_{\alpha'\beta'} = F_{\alpha\beta} \left[D^{\alpha\beta mn} D^{mn}_{\alpha'\beta'} \right] G_{mn}$$

Let

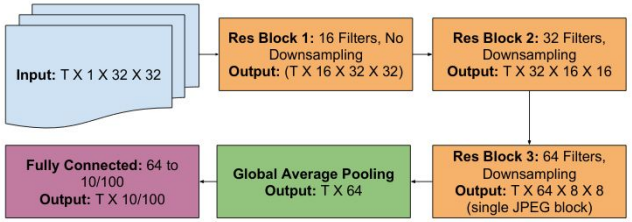
$$H^{\alpha\beta mn}_{\alpha'\beta'} = D^{\alpha\beta mn} D^{mn}_{\alpha'\beta'}$$

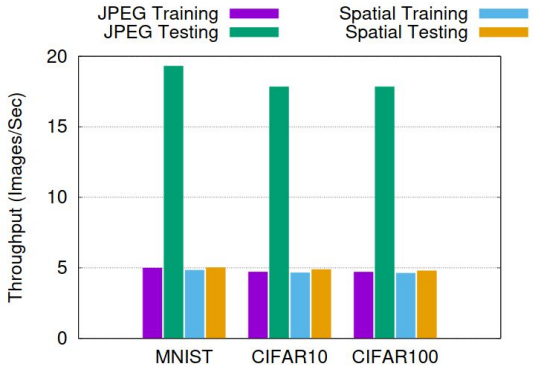
Then

$$F'_{\alpha'\beta'} = F_{\alpha\beta} H^{\alpha\beta mn}_{\alpha'\beta'} G_{mn}$$

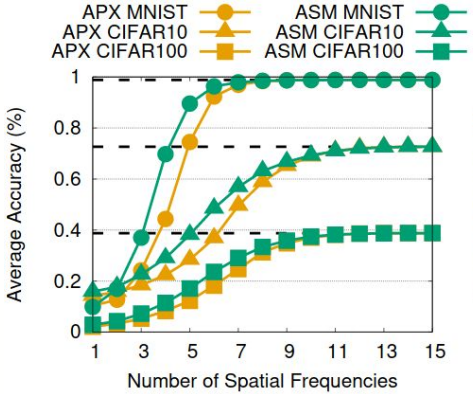
Results

- Used a toy model with MNIST, CIFAR-10,100
- Examined ReLU error and throughput
  - Much higher throughput for inference
  - Slightly higher throughput for training

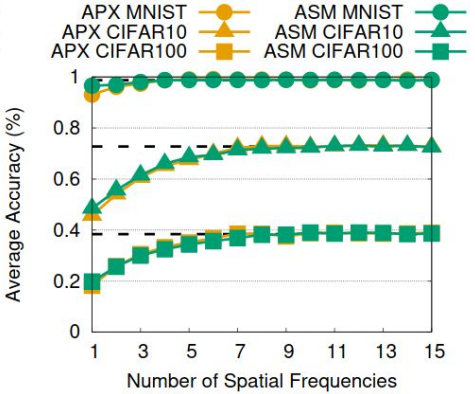




Throughput



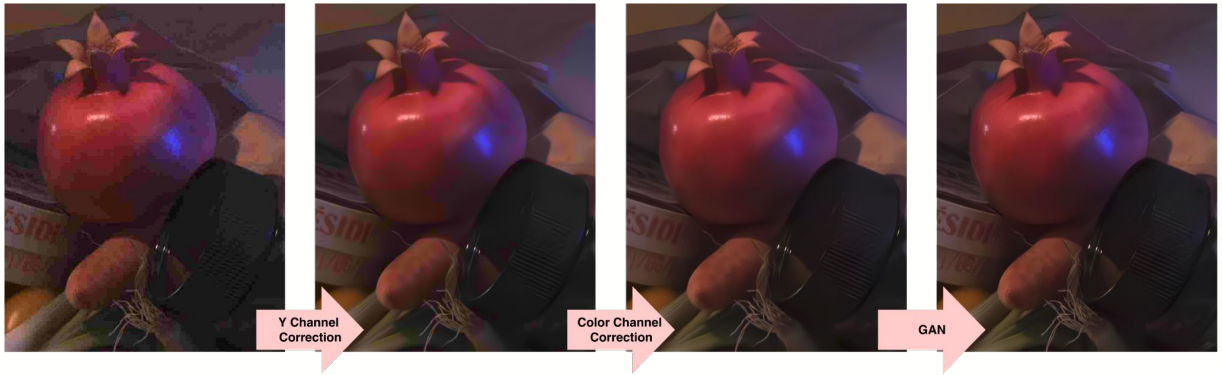
ReLU Accuracy
Model Conversion



ReLU Accuracy
Retraining

Quantization Guided JPEG Artifact Correction

- Goal: Compress and image at low quality, design a neural network to restore the image
- Allows for low bandwidth transmission of the image while still looking presentable
- Three major issues with prior work that make this impractical
  - Prior work is quality specific
  - Prior work deals with grayscale images only
  - Prior work uses standard error based regression



Solution - First Principles

Problems

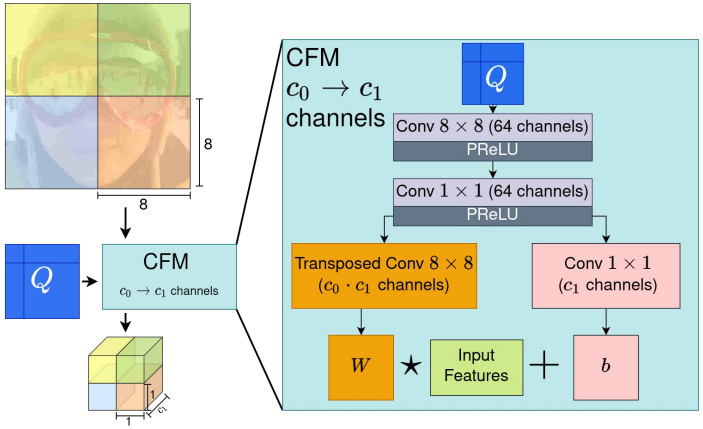
- Quality dependent models
  - Impractical to train, quality not stored in JPEG
- Grayscale only models
  - Grayscale model is easier to restore, but not applicable to color images
- Error based regression
  - Produces a blurry result

Solutions

- Use the quantization matrix (which is stored) as side data to a single network
- Color channels are usually subsampled and quantized at lower quality.
  - Restore grayscale channel first
  - Use restored grayscale channel to guide restoration of color channels
- Formulate a GAN loss which specifically restores texture

Quantization Side Data

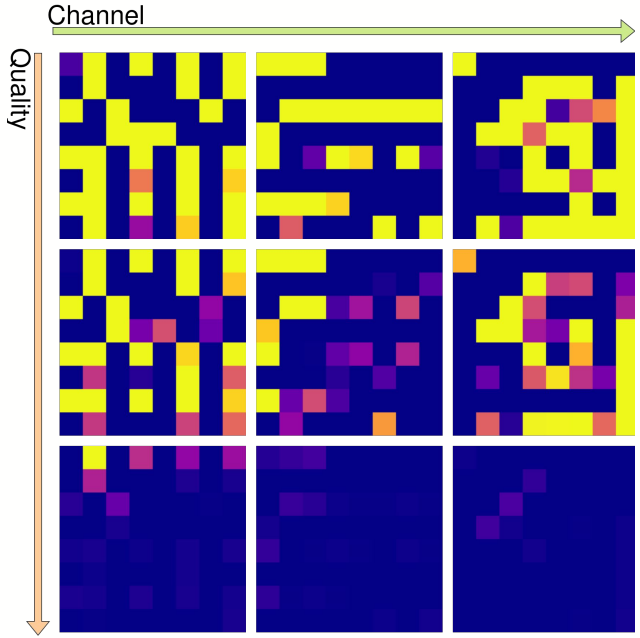
- We develop a technique called Convolutional Filter Manifolds an extension of Filter Manifolds [1]
- This is a general technique that uses a lightweight CNN to produce a convolutional filter for the network to apply to DCT coefficients

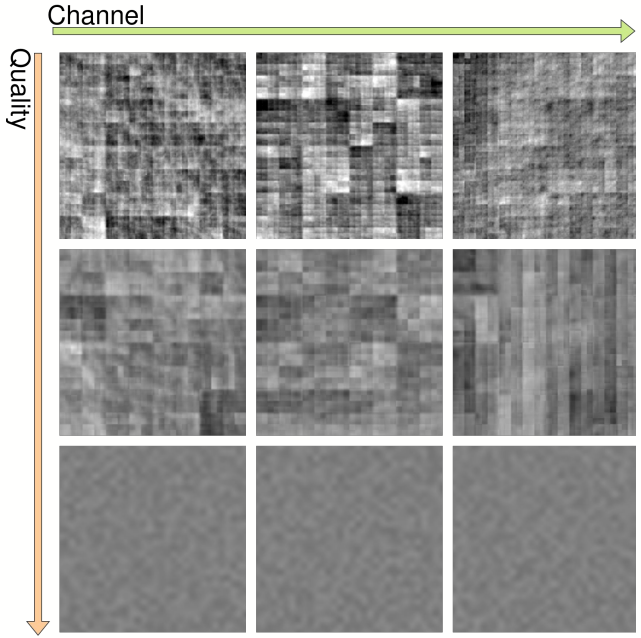


1. Kang, Di, Debarun Dhar, and Antoni B. Chan. "Crowd counting by adapting convolutional neural networks with side information." arXiv preprint arXiv:1611.06748 (2016).

CFMs Do Cool Things (1)

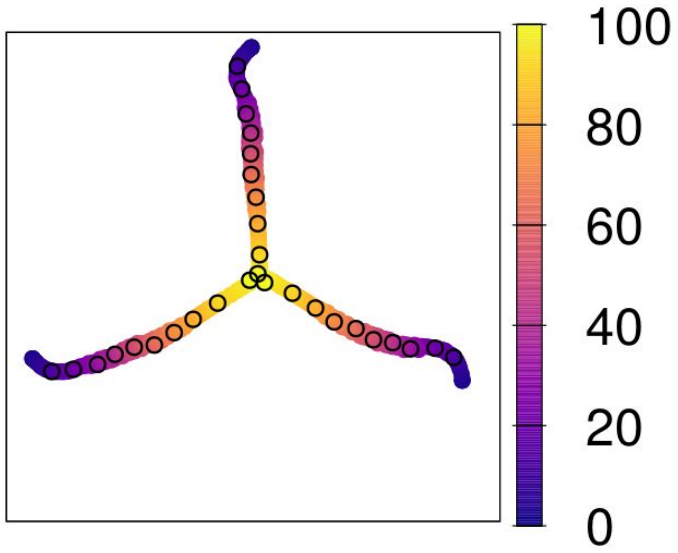
Take a single CFM layer, generate the weight for a quality 10, 50, 100 quantization matrix. Take three channels.





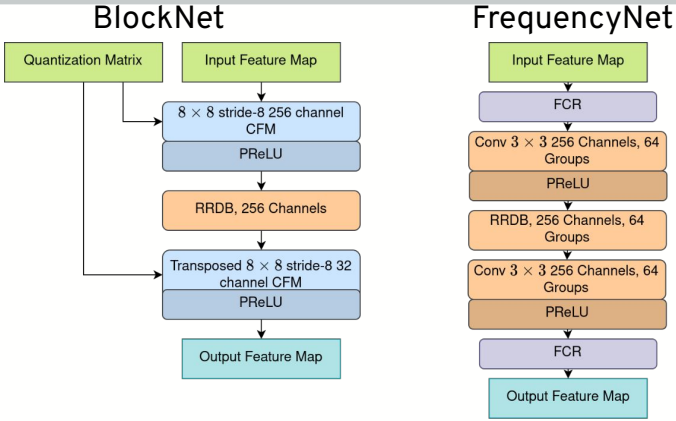
CFMs Do Cool Things (2)

- Now take all the quantization matrices, 0-100.
- Generate the weights and t-SNE to 2D. Then plot three channels with the color showing the quality integer that generated the weight.
- Not only are the filters well separated in space, but they are ordered according to their quality
- They spiral inwards to a point in the center at quality 100 where they all aren't doing much
- Circles are quantization matrices seen during training

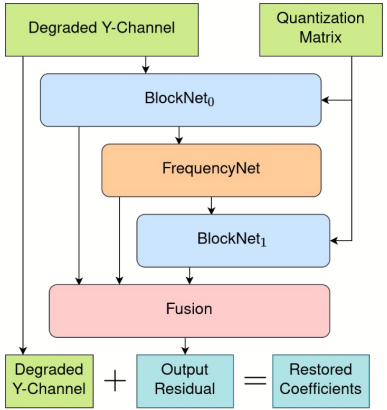


Grayscale Network

- We formulate coefficients-to-coefficients regression
  - This is important for using the quantization matrix because it directly describes rounding applied to DCT coefficients not pixels
- This requires some tricks, we use both
  - BlockNet [1]: uses 8 by 8 stride-8 convolutions to compute a representation of each DCT block
  - FrequencyNet [2]: rearranges the coefficients channel-wise and uses grouped convolutions to process frequencies in isolation



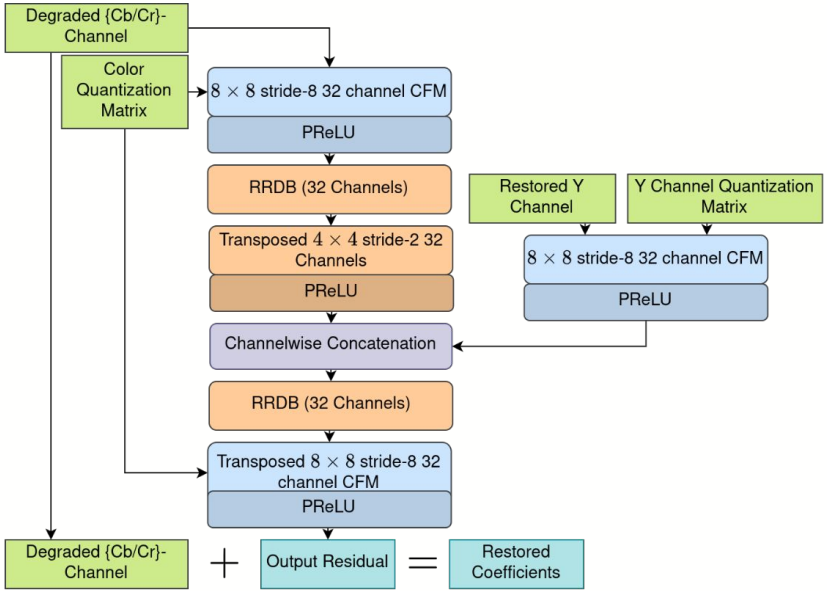
Full Y-Network



1. Deguerre, Benjamin, Clément Chatelain, and Gilles Gasso. "Fast object detection in compressed jpeg images." 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE, 2019.
2. Lo, Shao-Yuan, and Hsueh-Ming Hang. "Exploring semantic segmentation on the DCT representation." Proceedings of the ACM Multimedia Asia. 2019. 1-6.

Color Network

- Color images are usually chroma subsampled and compressed at lower quality
- This destroys much of the structure in the images
- Use the restored grayscale channel to provide that structure
- Includes a learned upsampling layer to undo the subsampling



GAN Texture Loss

- Regression alone produces a blurry result, so we introduce a GAN loss designed to restore texture
- Full GAN loss (given network output x and ground truth y)

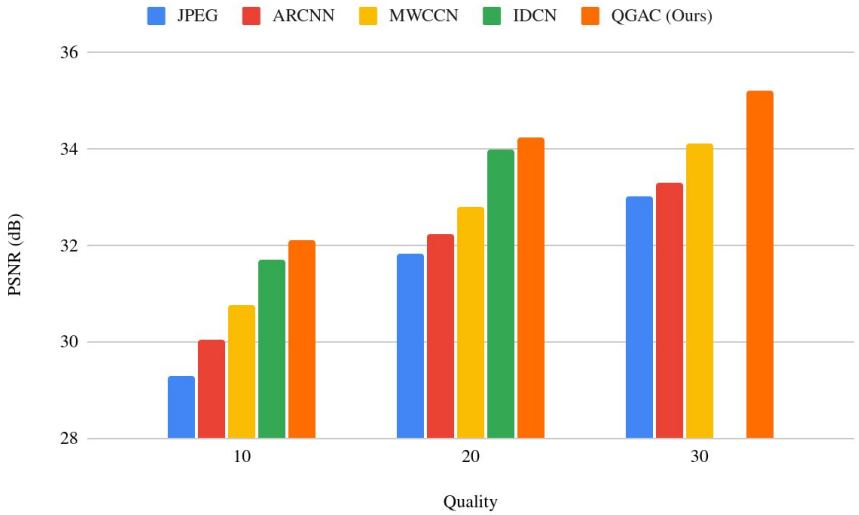
$$\mathcal{L}_{\text{GAN}} = \mathcal{L}_{\text{texture}}(x, y) + \gamma \mathcal{L}_G^{\text{Ra}} + \nu \|x - y\|_1$$

- Texture term uses a VGG network trained on the MINC materials dataset instead of an imagenet VGG perceptual loss
  - If the networks produce similar activations, they would be classified as the same material and therefore have similar texture

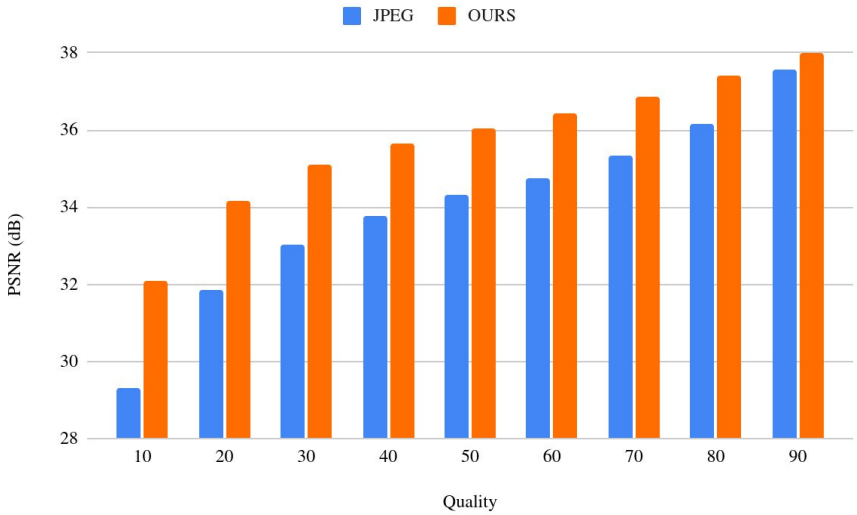
$$\mathcal{L}_{\text{texture}} = \|\text{MINC}_{5,3}(y) - \text{MINC}_{5,3}(x)\|_1$$

Numeric Results

Ours vs. Prior Work*



Ours Over Wide Quality Range



*See paper for citations

JPEG

Regression

GAN

JPEG

Restored

JPEG

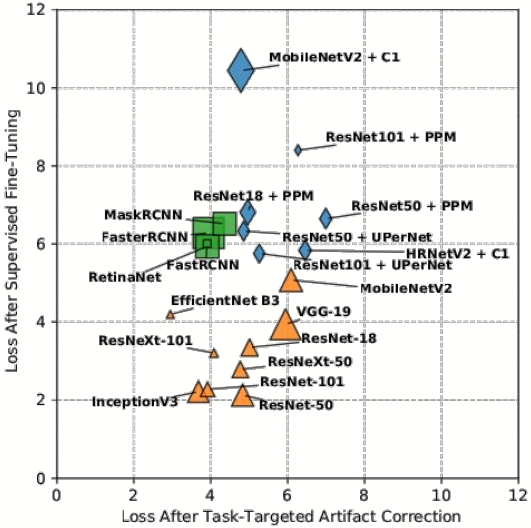
Restored

JPEG

Restored

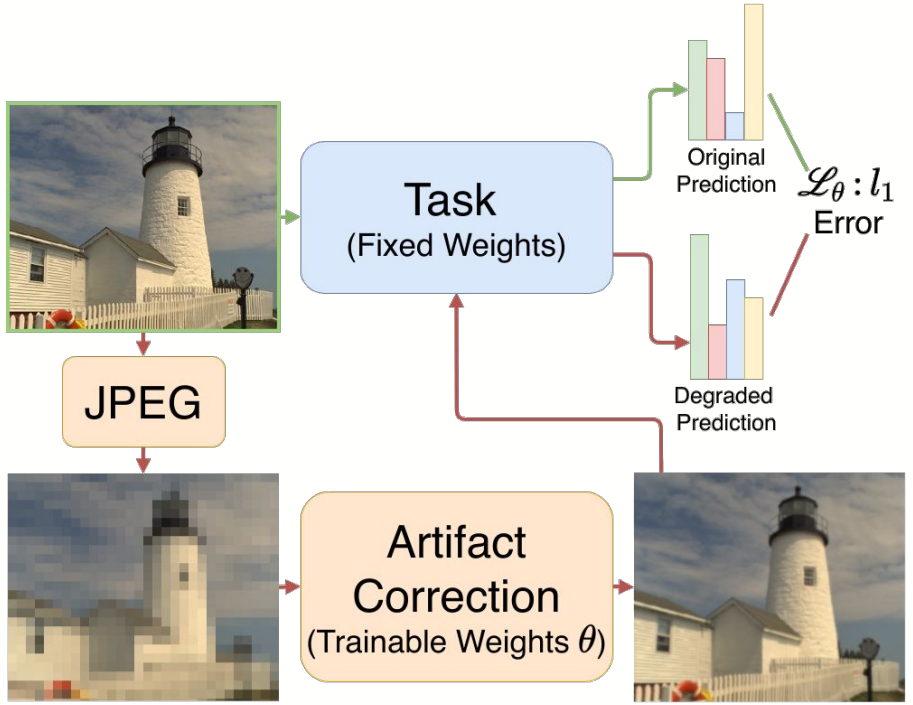
Analyzing and Mitigating JPEG Compression Defects in Deep Learning

- We talked about restoration for human consumption, what about machine consumption?
  - As deep models are increasingly deployed in consumer settings, they increasingly encounter compressed data
  - Academic datasets are lightly compressed, if at all
- How do deep models cope with this quality loss?
  - Do the lose performance on JPEG inputs?
  - **Can anything be done to mitigate the performance loss?**
  - We conducted a large scale study to answer these questions
- Best way to mitigate depends on the task
  - Global tasks (classification) do better with simple fine-tuning
  - Localization tasks (segmentation) do better with **task-targeted artifact correction**



Task-Targeted Artifact Correction

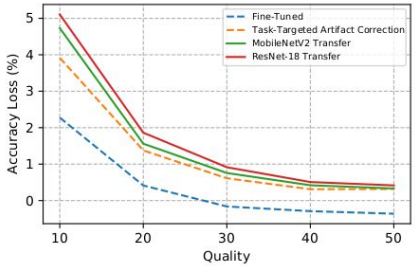
- Use loss from a downstream task network to fine-tune an artifact correction network
- This allows the AC network to focus on artifacts that degrade the prediction of the task network
- Completely self-supervised, no labels required to train (only uses logits)
- Supports transfer where an AC network trained for one task is used for another one
- Supports multihead where multiple task networks are used for training



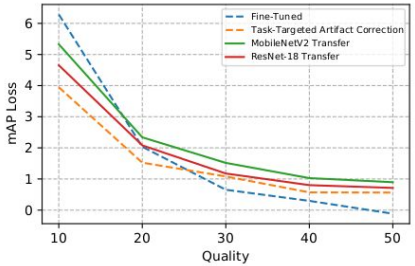
Transfer and Multihead

Transfer

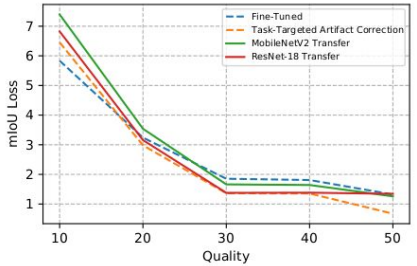
Resnet-101



FasterRCNN

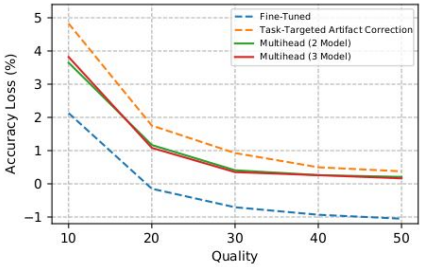


HRNetV2 + C1

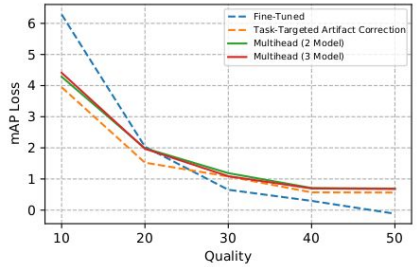


Multihead

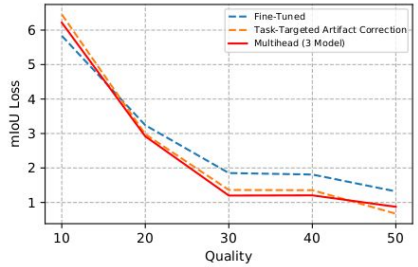
Resnet-50



FasterRCNN



HRNetV2 + C1

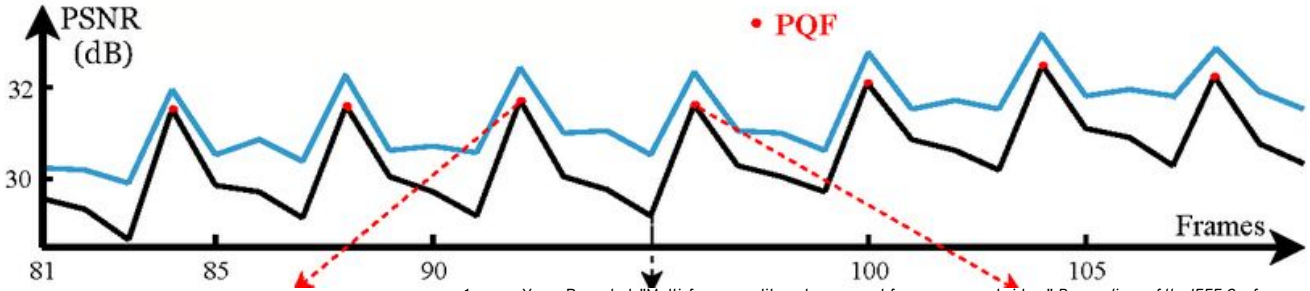


MetaBit

- DL for Video Compression is making progress
  - Latest work from Wave 1 can (sort of) match AV1 for rate-distortion, but significantly slower
  - Not going to be competitive/consumer ready any time soon
- **How do we get DL into video compression in the near term?**
  - Keep H.264 around!
- Brings big advantages
  - People without GPUs can view the resulting stream (it's just H.264)
  - Encode time: 60+fps on a CPU (i.e., no DL required…)
  - Decode time: faster than many DL codecs (could be real-time with some elbow grease)
  - Rate-distortion: better than many DL codecs (could be better with more training)
  - Add a GAN loss: looks amazing (but temporal consistency issues)

Not Quite a New Idea

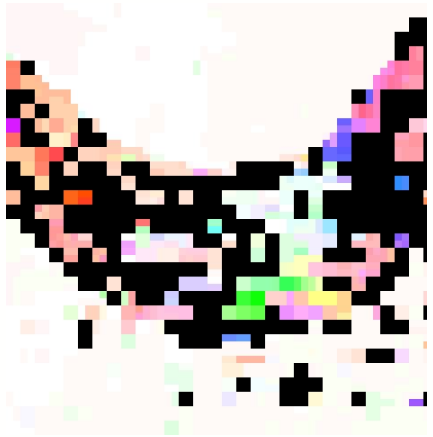
- Prior work treated this as multiframe “image-to-image” regression
- Leveraged “standard procedures” from restoration tasks like super-resolution
  - Optical flow for alignment[1], or deformable convolutions [2]
- One unique idea: Peak Quality Frames [1]
  - Train a model to detect frames which are high quality, use those to guide restoration of the low quality frames



1. Yang, Ren, et al. "Multi-frame quality enhancement for compressed video." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
2. Deng, Jianing, et al. "Spatio-temporal deformable convolution for compressed video quality enhancement." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 07. 2020.

How Does Video Compression Work?

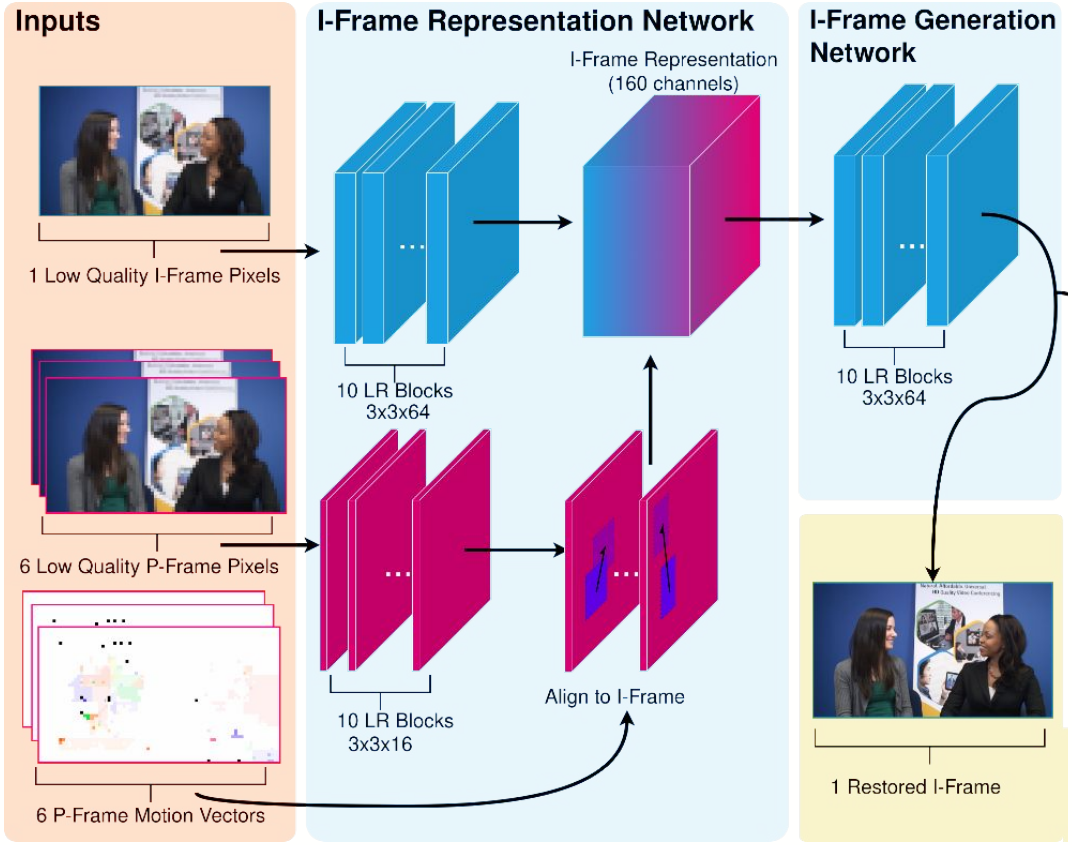
- Entropy reduction in space (like JPEG) and **time**
  - **I-Frames**: (intra-frames) like a JPEG image, are decoded with no external information
  - **P-Frames:** (predicted frames) Require at least one prior frame to decode
  - **GOP:** (group of pictures) I-frame and its associated P-frames
- I-frames are straightforward, what about P-Frames?
  - Motion Vectors: block based heuristic motion estimation
  - Error residuals: everything that can't be modeled by the motion vectors (i.e., compute estimated frame from motion vectors, store difference (residual) between true and predicted)

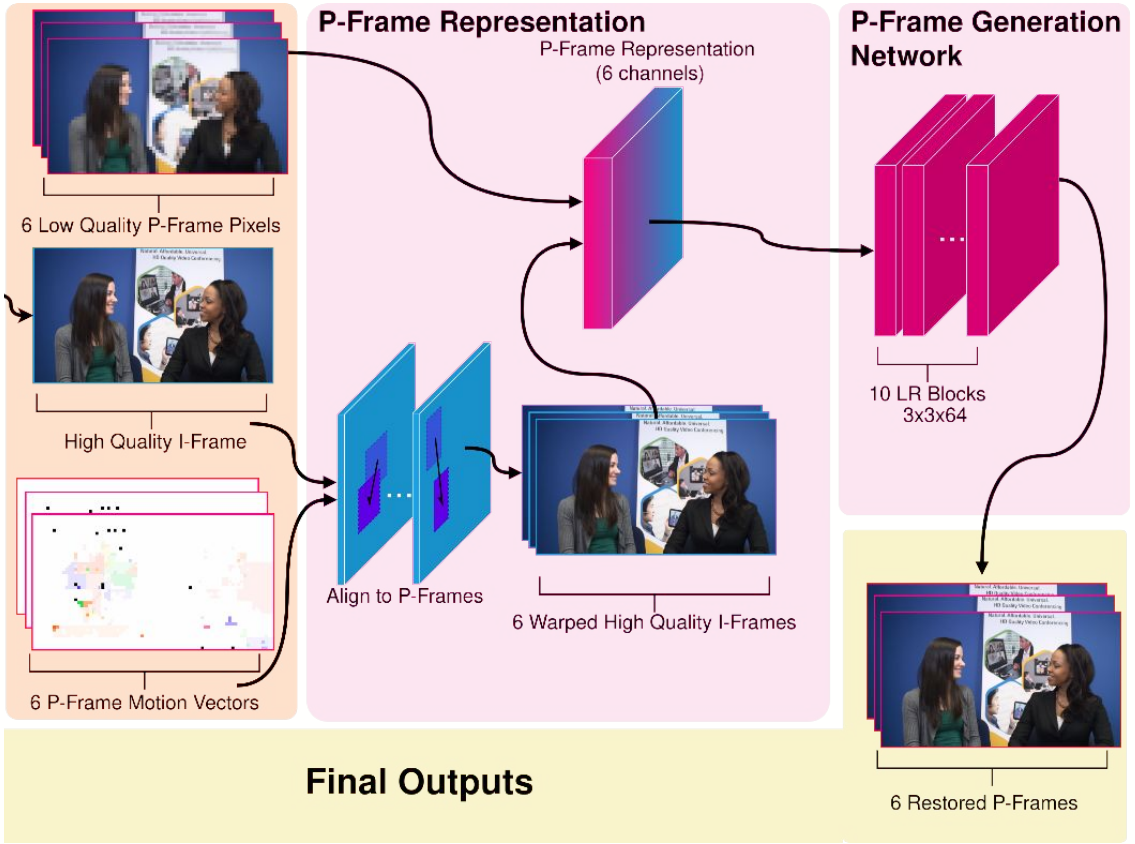
We Can Use This Knowledge! - First Principles!

- Motion vectors: coarse motion estimation
  - Don’t need optical flow
- I-Frames: Are necessarily stored at higher quality
  - Don’t need to search for PQFs
- Our Network
  - Reads motion vectors and aligns feature maps (Fast!)
  - Use I-Frame restorations to restore P-frames (Accurate!)
  - Reads GOP structure to restore blocks of frames (Fast!)
- We saved parameters by not needing optical flow/PQF detection
  - So re-invest those parameters in the restoration network (Even more accurate!)
- Plus some fun loss formulations

Phase 1 - Do A Really Good Job on the I Frame



Phase 2 - Use the Really Good I Frame To Restore the P Frame



Fun with Loss Functions!

- Compression specifically targets high frequency details
  - But: traditional regression loss functions encourage “averaging” which implicitly prioritizes low frequencies
  - Force the network to weight frequency bands equally using a difference-of-gaussians scale-space loss
  - Given $G(\sigma)$ gaussian kernel with std. dev sigma and O_s as the network output downsampled by a factor of s

Compute Filtered Images

$$
\begin{aligned}
I_{O,s,1} &= G(1.1) * O_s \\
I_{O,s,2} &= G(2.2) * O_s \\
I_{O,s,3} &= G(3.3) * O_s \\
I_{O,s,4} &= G(4.4) * O_s
\end{aligned}
$$

Compute DoG

$$
\begin{aligned}
B_{O,s,1} &= I_{O,s,2} - I_{O,s,1} \\
B_{O,s,2} &= I_{O,s,3} - I_{O,s,2} \\
B_{O,s,3} &= I_{O,s,4} - I_{O,s,3}
\end{aligned}
$$

- Final Loss Function: $\mathcal{L}_{\text{DoG}}(O,T) = \sum_{s \in \{1,2,4,8\}} \sum_{b=1}^{3} ||B_{T,s,b} - B_{O,s,b}||_1$

Fun with Loss Functions! Part 2

- Regression losses are great for numerical results, not so much for looking at
- Add a GAN and texture loss to make the images look great
- Wasserstein GAN, critic augmented with temporal consistency (TeCoGAN [1])
- Texture loss from my JPEG work

Regression Loss (numerical results)

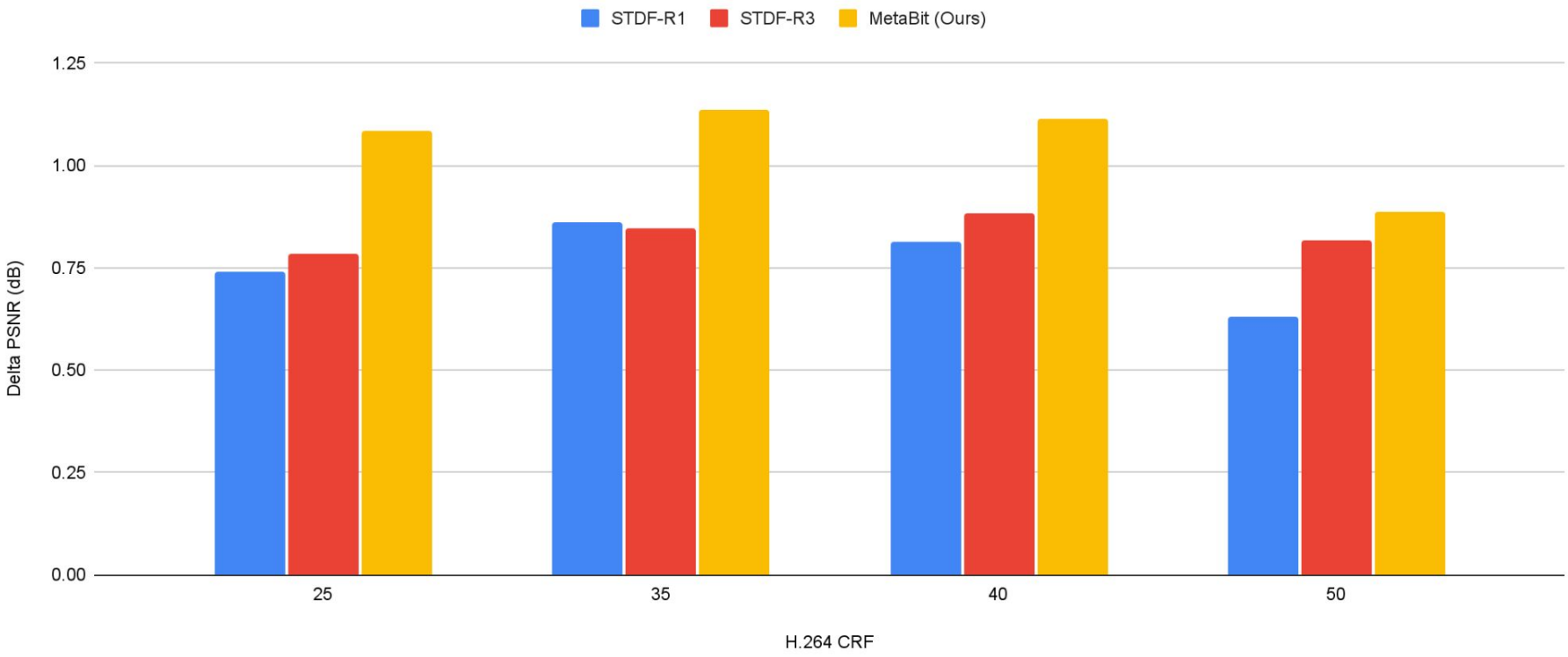
$$\mathcal{L}_R = \alpha\mathcal{L}_1(O,T) + \beta\mathcal{L}_{\text{DoG}}(O,T)$$

GAN Loss (qual. results)

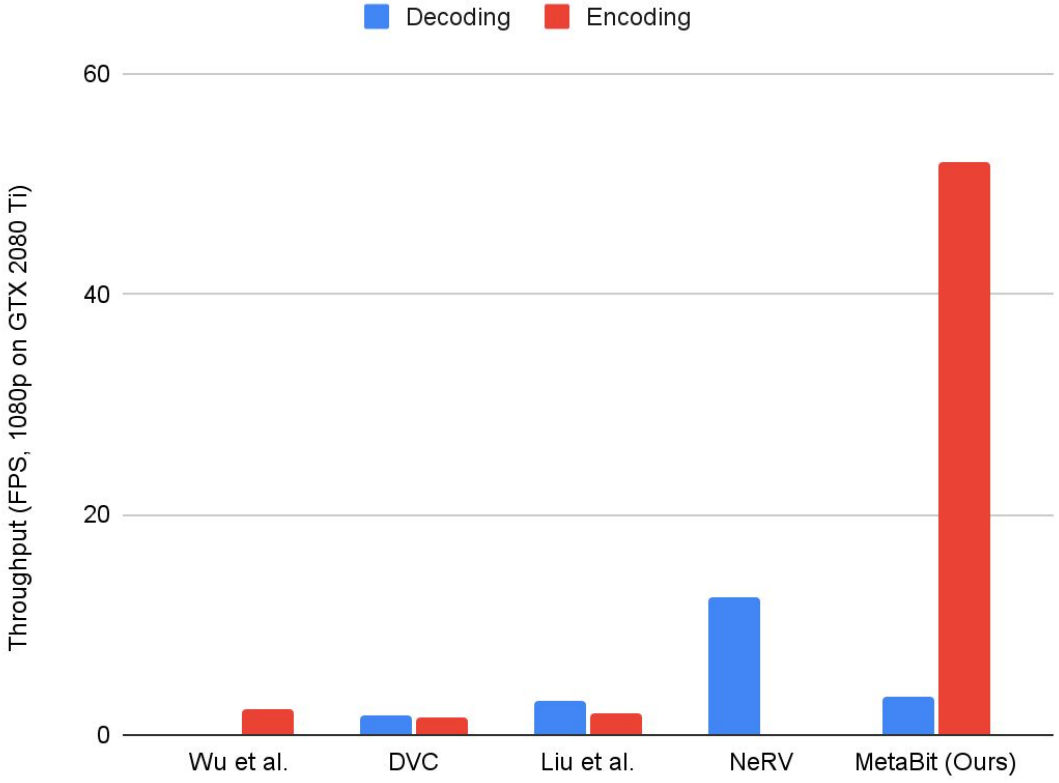
$$\mathcal{L}_{\text{GAN}}(O,T) = \alpha\mathcal{L}_1(O,T) + \beta\mathcal{L}_{\text{DoG}}(O,T) + \gamma\mathcal{L}_{\text{W}}(O,T) + \delta\mathcal{L}_{\text{texture}}(O,T)$$

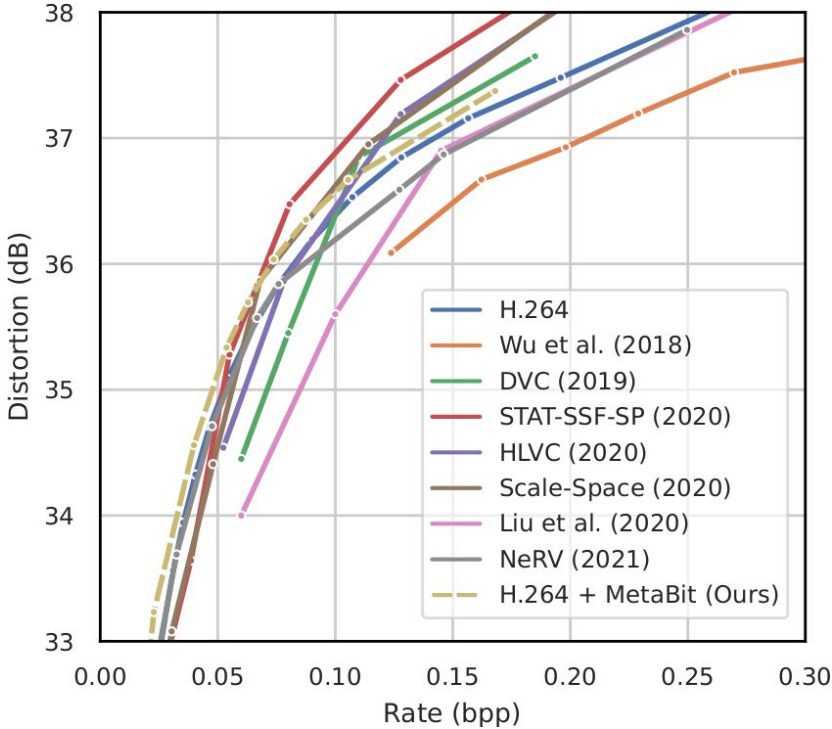
1. Chu, Mengyu, et al. "Temporally coherent gans for video super-resolution (tecogan)." arXiv preprint arXiv:1811.09393 1.2 (2018)

Quantitative Results



Comparison to DL Compression






Cal National
110 SOUTH
San Pedro
SOUTH
6th St
Wilshire Blvd


Cal National
110 SOUTH
San Pedro
110 SOUTH
6th St
Wilshire Blvd







Recap

- Compression is important, it powers the modern internet, and allows more people to participate
- Deep learning can help produce smaller images/videos
  - But we're a long way off from practical compression powered solely by deep learning
- By working **with** classical compression, we can realize improved compression in the **near term**
  - And solve some fun related problems
- The Future
  - Faster, better algorithms
  - New problems (data security, better frequency aware models, etc.)
  - Consumer applications?