Recent Progress on CNNs for Object Detection & Image Compression

Rahul Sukthankar
Google Research

Credits: My Research Group at Google

Lifelong Learning
- Vitto Ferrari (TL)
- Danfeng Qin
- Hassan Rom
- Jasper Uijlings
- Stefan Popov

3D People/VR/AR
- Chris Bregler (TL)
- Avneesh Sud
- Christian Frueh
- Diego Ruspini
- Nick Dufour
- Nori Kanazawa
- Vivek Kwatra

Object Detection ++
- Kevin Murphy (TL)
- Alireza Fathi
- Anoop Korattikara
- Chen Sun
- George Papandreou
- Hyun Oh Song
- Jonathan Huang
- Nathan Silberman
- Sergio Guadarrama
- Tyler Zhu
- Vivek Rathod

Learning from Video
- Susanna Ricco (TL)
- Alexey Vorobyov
- Bryan Seybold
- Dave Marwood
- David Ross
- Sudheendra Vijayanarasimhan

Event Understanding
- Caroline Pantofaru (TL)
- Arthur Wait
- Cheol Park
- Eric Nichols
- Radhika Marvin
- Shrenik Lad
- Vinay Bettadapura

NN Compression
- George Toderici (TL)
- Damien Vincent
- David Minnen
- Joel Shor
- Nick Johnston
- Michele Covell
- Saurabh Singh
- Sung Jin Hwang

NN Theorem Proving
- Christian Szegedy (TL)
- Alex Alemi
- Niklas Een
- Sarah Loos

Individual Explorers
- Chunhui Gu
- Ian Fischer
- Mohamad Tarifi
- Noah Snavely
- Shumeet Baluja

Part-Time Faculty
- Abhinav Gupta
- Irfan Essa
- Jitendra Malik
- Kate Fragkiadaki

[+ Noah & Vitto]

Credits: My Research Group at Google

Lifelong Learning

- Vitto Ferrari (TL)
- Danfeng Qin
- Hassan Rom
- Jasper Uijlings
- Stefan Popov

3D People/VR/AR

- Chris Bregler (TL)
- Avneesh Sud
- Christian Frueh
- Diego Ruspini
- Nick Dufour
- Nori Kanazawa
- Vivek Kwatra

Object Detection ++

- Kevin Murphy (TL)
- Alireza Fathi
- Anoop Korattikara
- Chen Sun
- George Papandreou
- Hyun Oh Song
- Jonathan Huang
- Nathan Silberman
- Sergio Guadarrama
- Tyler Zhu
- Vivek Rathod

Part 1

Learning from Video

- Susanna Ricco (TL)
- Alexey Vorobyov
- Bryan Seybold
- Dave Marwood
- David Ross
- Sudheendra Vijayanarasimhan

Event Understanding

- Caroline Pantofaru (TL)
- Arthur Wait
- Cheol Park
- Eric Nichols
- Radhika Marvin
- Shrenik Lad
- Vinay Bettadapura

NN Compression

- George Toderici (TL)
- Damien Vincent
- David Minnen
- Joel Shor
- Nick Johnston
- Michele Covell
- Saurabh Singh
- Sung Jin Hwang

NN Theorem Proving

- Christian Szegedy (TL)
- Alex Alemi
- Niklas Een
- Sarah Loos

Individual Explorers

- Chunhui Gu
- Ian Fischer
- Mohamad Tarifi
- Noah Snavely
- Shumeet Baluja

Part-Time Faculty

- Abhinav Gupta
- Irfan Essa
- Jitendra Malik
- Kate Fragkiadaki

[+ Noah & Vitto]

Credits: My Research Group at Google

Lifelong Learning

- Vitto Ferrari (TL)
- Danfeng Qin
- Hassan Rom
- Jasper Uijlings
- Stefan Popov

3D People/VR/AR

- Chris Bregler (TL)
- Avneesh Sud
- Christian Frueh
- Diego Ruspini
- Nick Dufour
- Nori Kanazawa
- Vivek Kwatra

Object Detection ++

- Kevin Murphy (TL)
- Alireza Fathi
- Anoop Korattikara
- Chen Sun
- George Papandreou
- Hyun Oh Song
- Jonathan Huang
- Nathan Silberman
- Sergio Guadarrama
- Tyler Zhu
- Vivek Rathod

Learning from Video

- Susanna Ricco (TL)
- Alexey Vorobyov
- Bryan Seybold
- Dave Marwood
- David Ross
- Sudheendra Vijayanarasimhan

Event Understanding

- Caroline Pantofaru (TL)
- Arthur Wait
- Cheol Park
- Eric Nichols
- Radhika Marvin
- Shrenik Lad
- Vinay Bettadapura

NN Compression

- George Toderici (TL)
- Damien Vincent
- David Minnen
- Joel Shor
- Nick Johnston
- Michele Covell
- Saurabh Singh
- Sung Jin Hwang

Part 2

NN Theorem Proving

- Christian Szegedy (TL)
- Alex Alemi
- Niklas Een
- Sarah Loos

Individual Explorers

- Chunhui Gu
- Ian Fischer
- Mohamad Tarifi
- Noah Snavely
- Shumeet Baluja

Part-Time Faculty

- Abhinav Gupta
- Irfan Essa
- Jitendra Malik
- Kate Fragkiadaki

[+ Noah & Vitto]

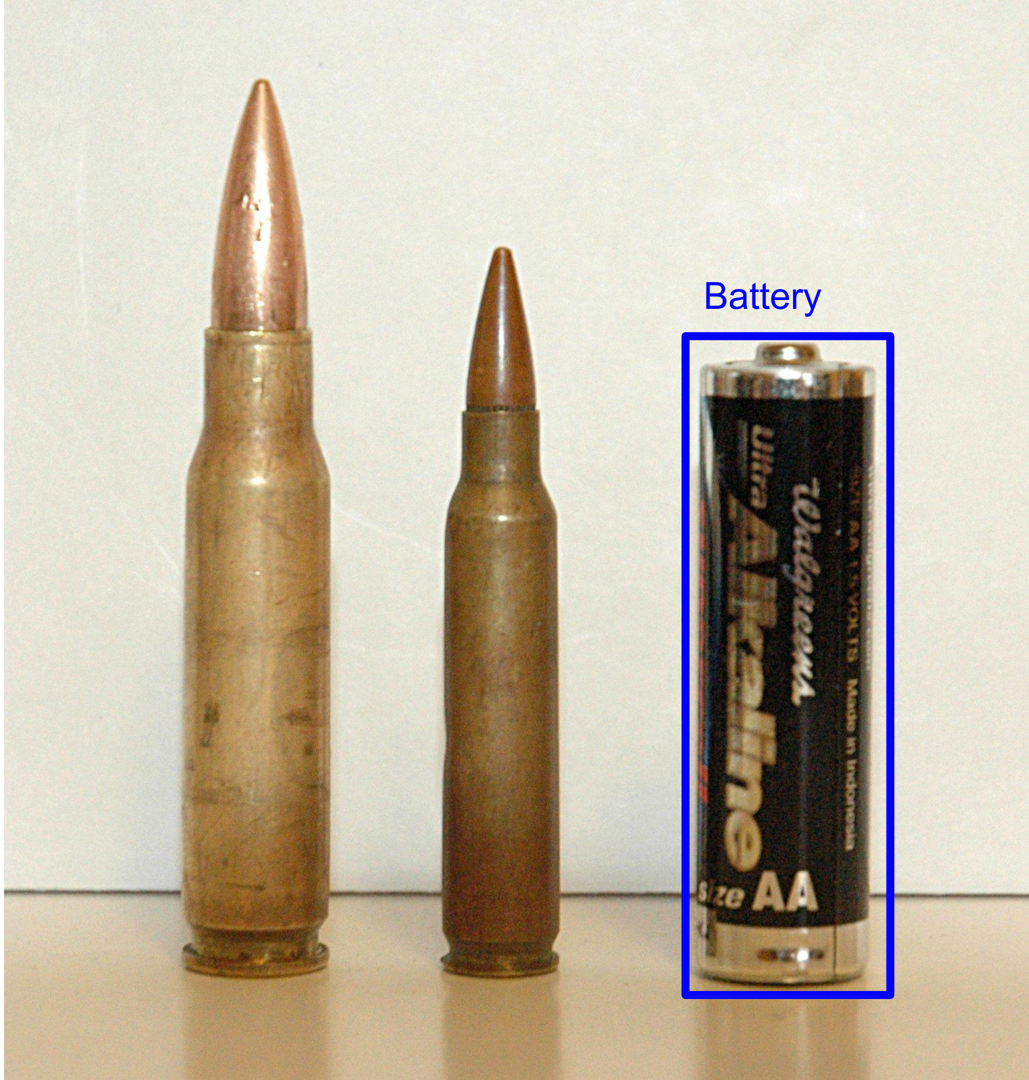
Part 1: Object Detection

Huang, Rathod, Sun, Zhu, Korattikara, Fathi, Fischer, Wojna, Song, Guadarrama, and Murphy,
“Speed/accuracy trade-offs for modern convolutional object detectors”
https://arxiv.org/abs/1611.10012

Object Detection

Object Detection

For a given set of object categories, mark each instance with a bounding box and a category label

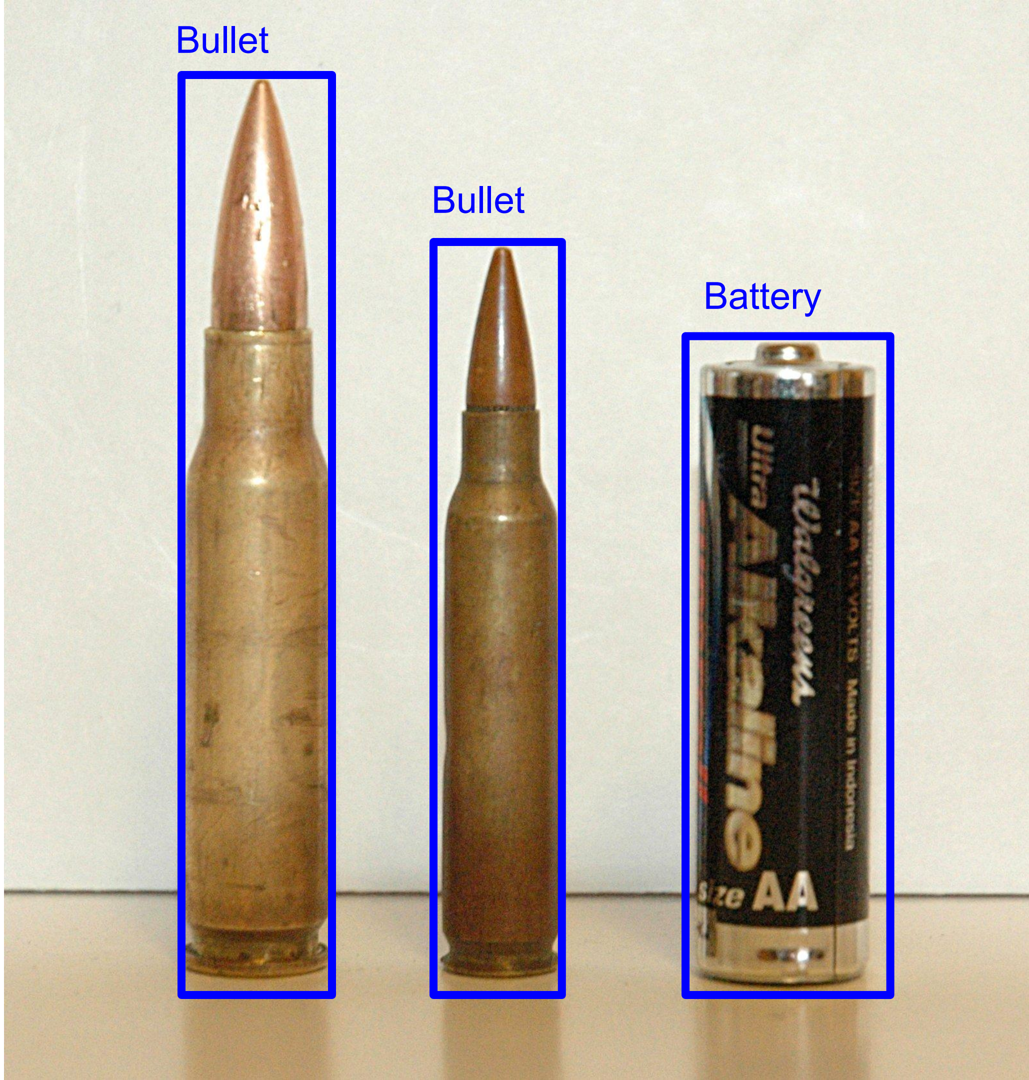


Google

Object Detection

For a given set of object categories, mark each instance with a bounding box and a category label

Can add object categories

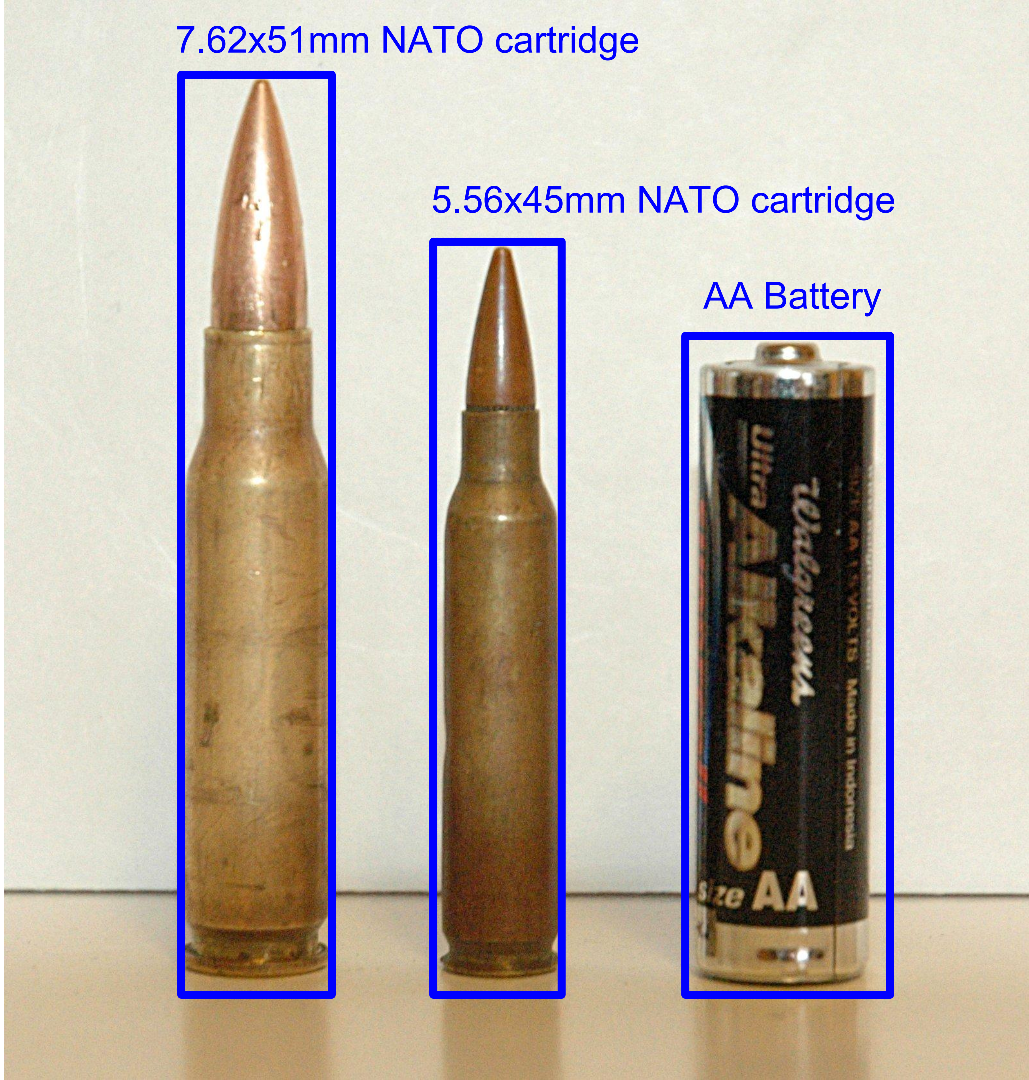


Google

Object Detection

For a given set of object categories, mark each instance with a bounding box and a category label

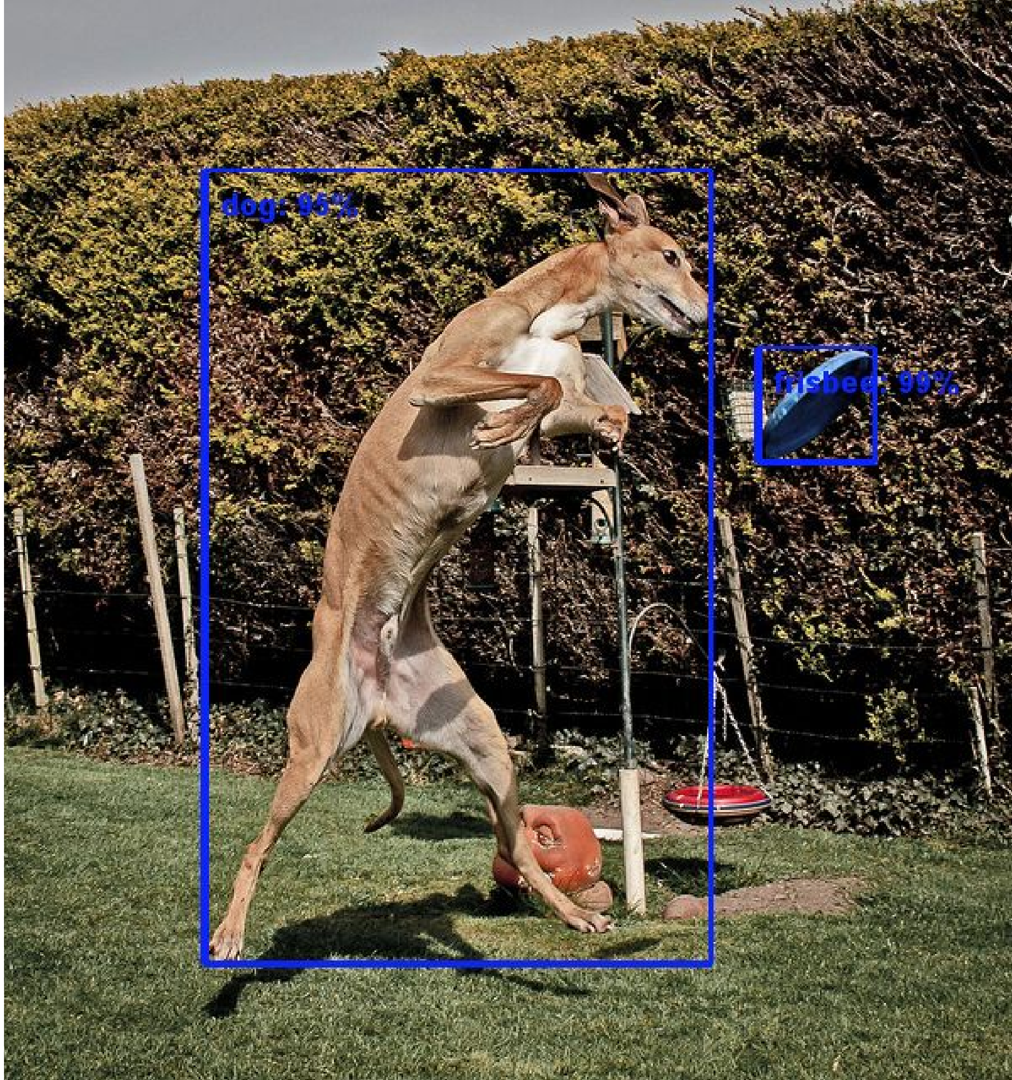
Can add *more* object categories (fine grained recognition)



Object Detection

For a given set of object categories, mark each instance with a bounding box and a category label

Becomes very challenging in complex scenes due to object size, clutter and partial occlusion

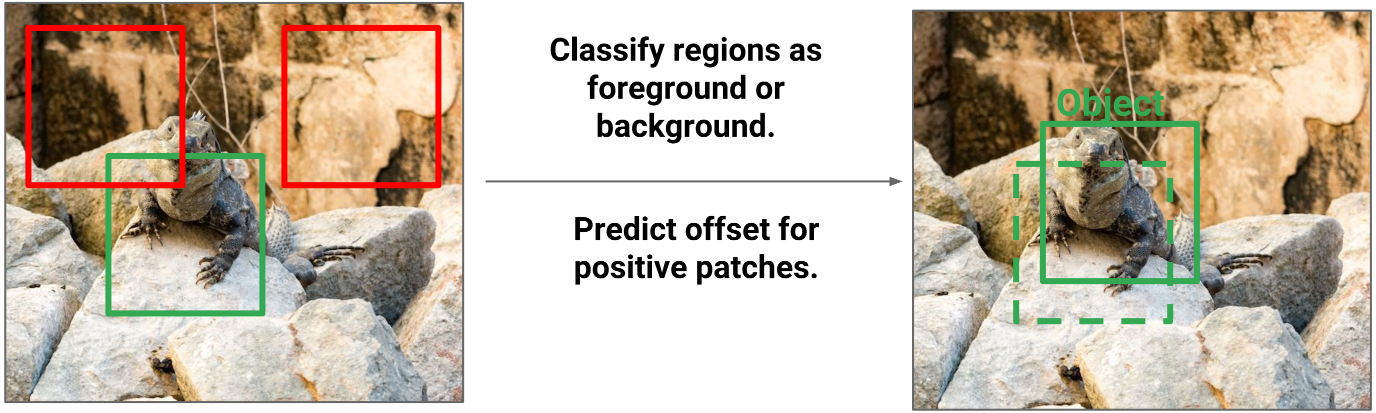


Google

Object Detection -- Sampling of Key Ideas

- Dense sliding windows -- searching over x, y, scale
- Neural net based face detection [Rowley et al., 1995]
- Classifier cascade, efficient ``integral image” features [Viola & Jones, 2001]
- HoG + SVM for pedestrian detection [Dalal & Triggs, 2005]
- Deformable part models [Felzenszwalb et al., 2010]
- Proposals (selective search) vs. sliding windows [e.g., van de Sande et al., 2011]
  {overcomes issue of densely sampling x, y, scale + aspect ratio}
- Return of neural nets -- learned feature extractors [Krizhevsky et al., 2012]
- Current generation of object detectors -- pioneered by Multibox and R-CNN.

Typical Modern Approach: Predict Region Offset & Classify

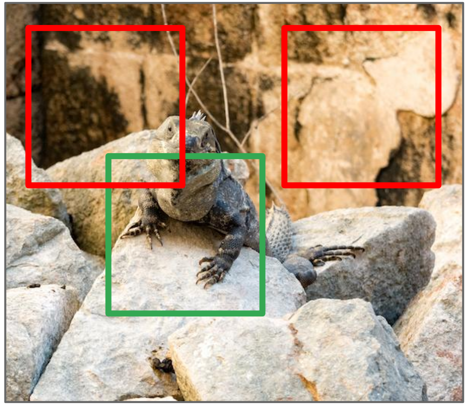


Classify regions as foreground or background.

Predict offset for positive patches.

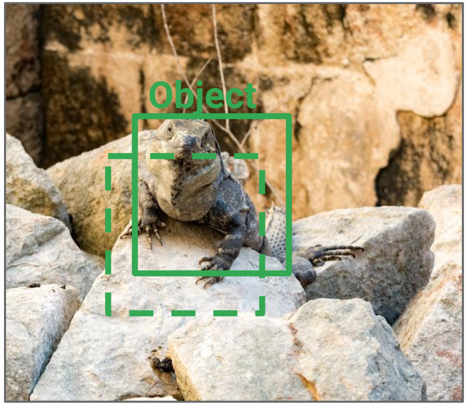
- Predicting bounding box offset is a counterintuitive concept
- How to select the initial boxes (often called *anchors*)?
  - External process (R-CNN)
  - Clustering ground truth boxes (Multibox)
  - Dense grid (now popular)
- Interesting connection to sliding windows and object proposals

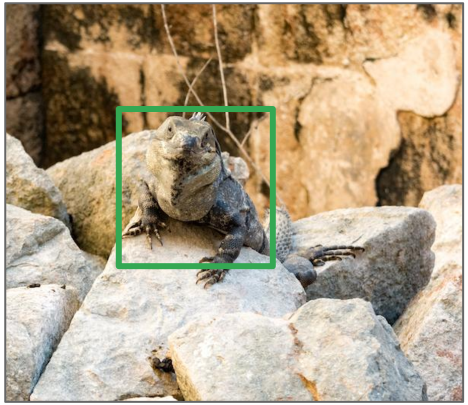
Typical Modern Approach: Predict Region Offset & Classify



Classify regions as foreground or background.

Predict offset for positive patches.

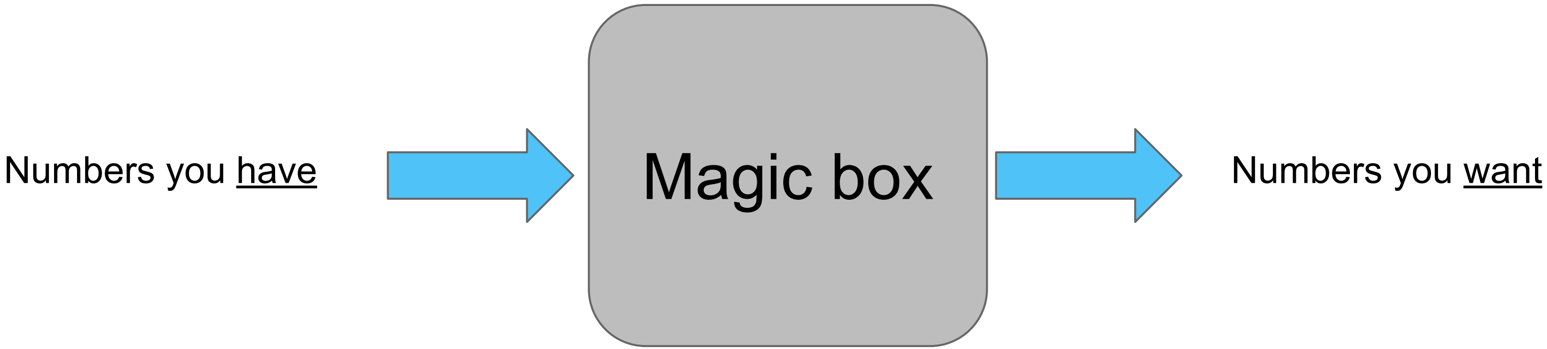




Classify foreground regions into 1 of C classes.

Lizard: 0.8
Frog: 0.1
Dog: 0.1

Aside: What is a Neural Network?



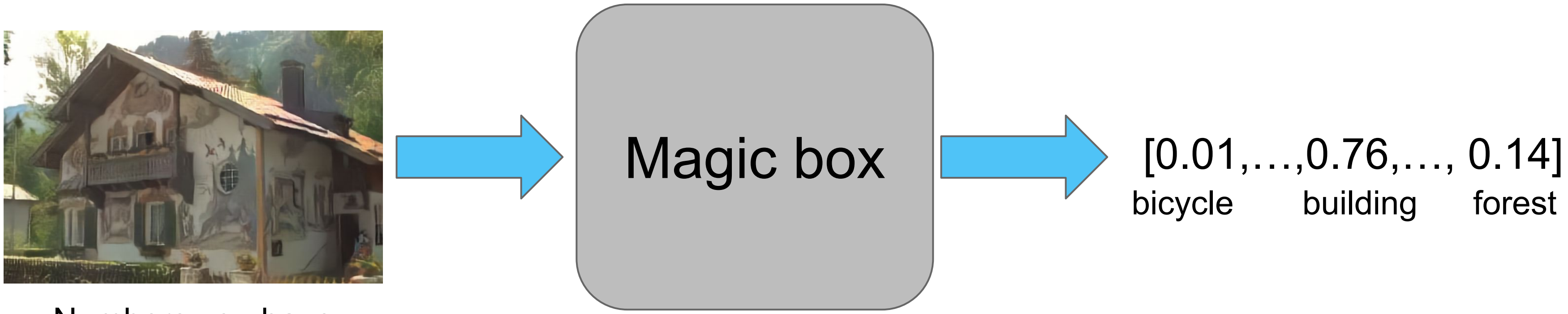
Numbers you have

Magic box

Numbers you want

Learns from lots of data using
gradient and grad student descent

Aside: What is a Neural Network?



Magic box

[0.01,…,0.76,…, 0.14]

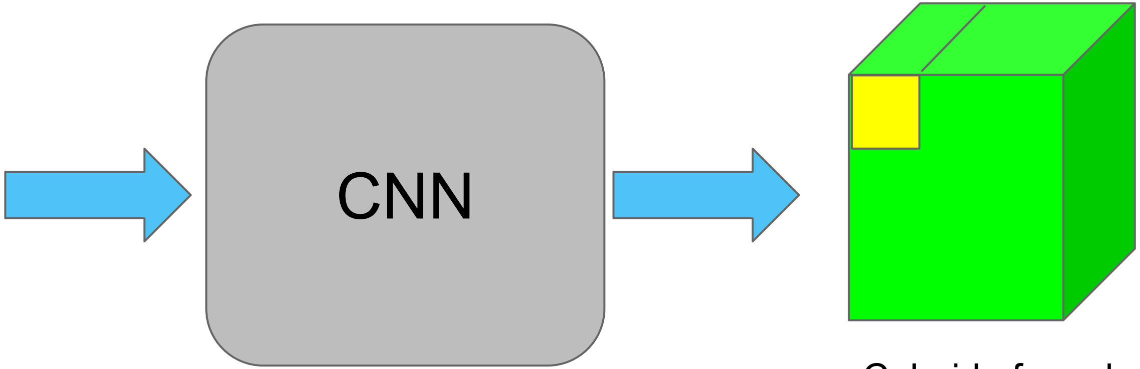
bicycle building forest

Numbers you have (e.g., RGB pixels)

Trained on a large labeled dataset like ImageNet

Aside: What is a Convolutional Neural Network?

Cuboid of numbers
(X x Y x D)



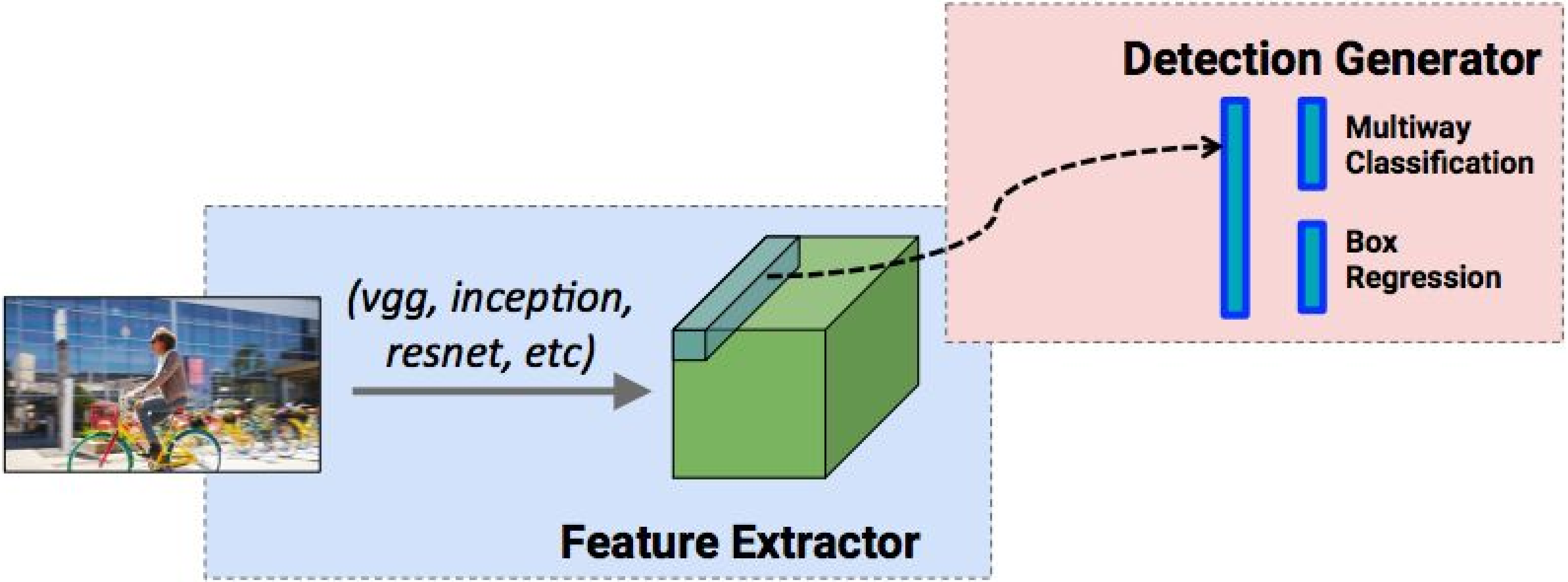
Cuboid of numbers
(X’ x Y’ x D’)

- Patch-to-patch mapping
- Shared weights (shift invariant)
- Retinal connectivity (local support)

Components of Modern Object Detection Systems

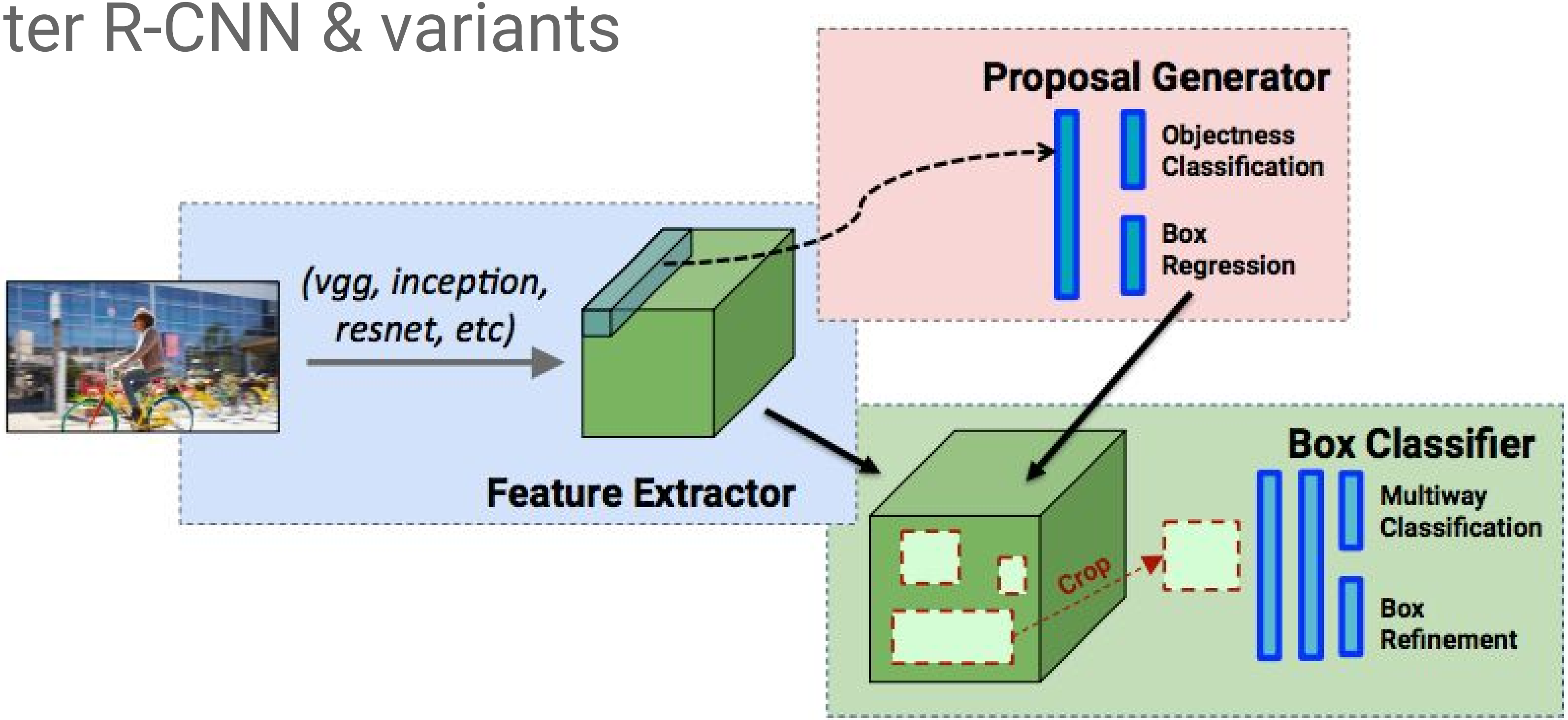
1. **Feature Extractor**
   Input: RGB pixels
   Output: a feature vector of numbers for each patch
2. **Proposal Generator**
   Input: feature vector
   Output: objectness classifier -- foreground or background?
   Output: bounding box regression -- where?
3. **Box Classifier** -- can be combined with (2)
   Input: features for cropped box
   Output: multi-way classifier -- what class is this object?
   Output: bounding box refinement -- how to adjust box to be on object

Object Detection Meta-Architecture Type 1: Single-Shot Detector (SSD) & variants



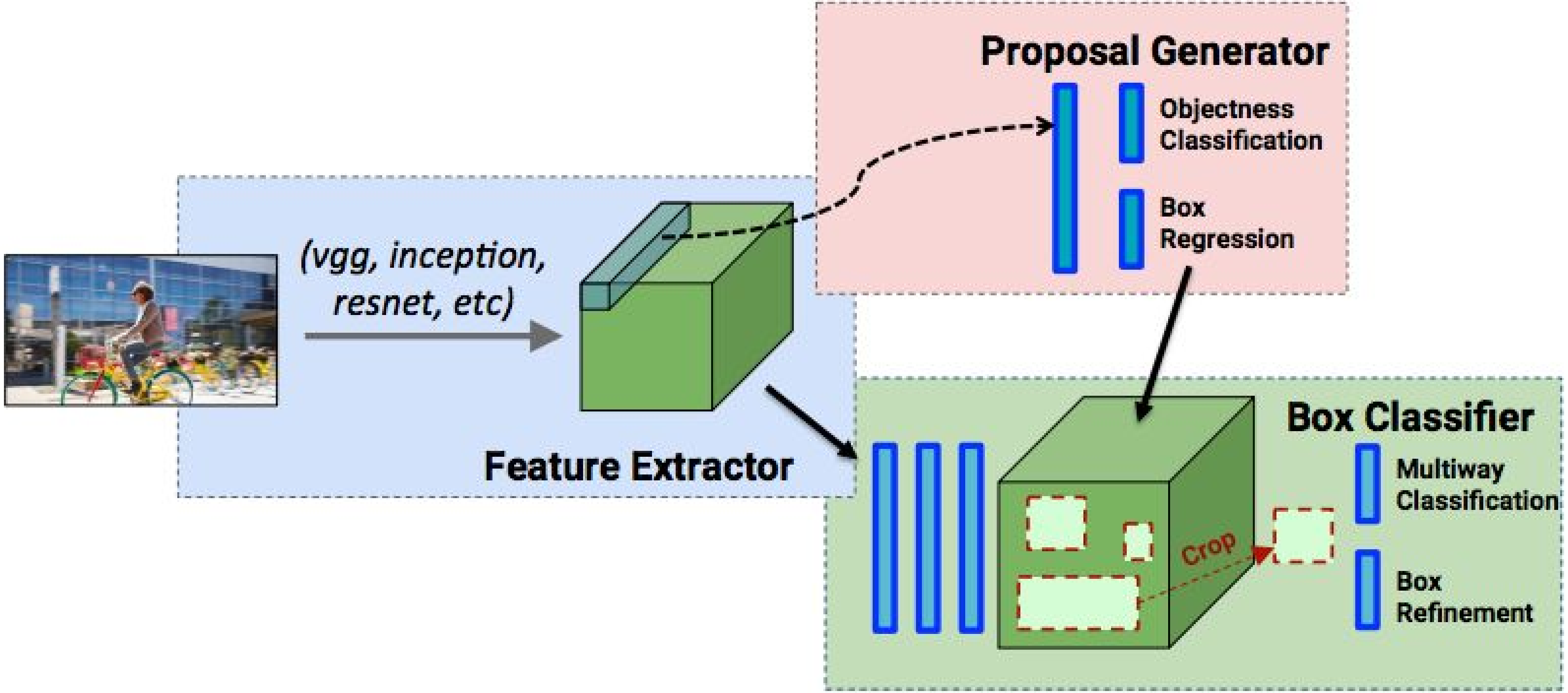
[Liu et al., 2015]

Object Detection Meta-Architecture Type 2: Faster R-CNN & variants



[Ren et al., 2015]

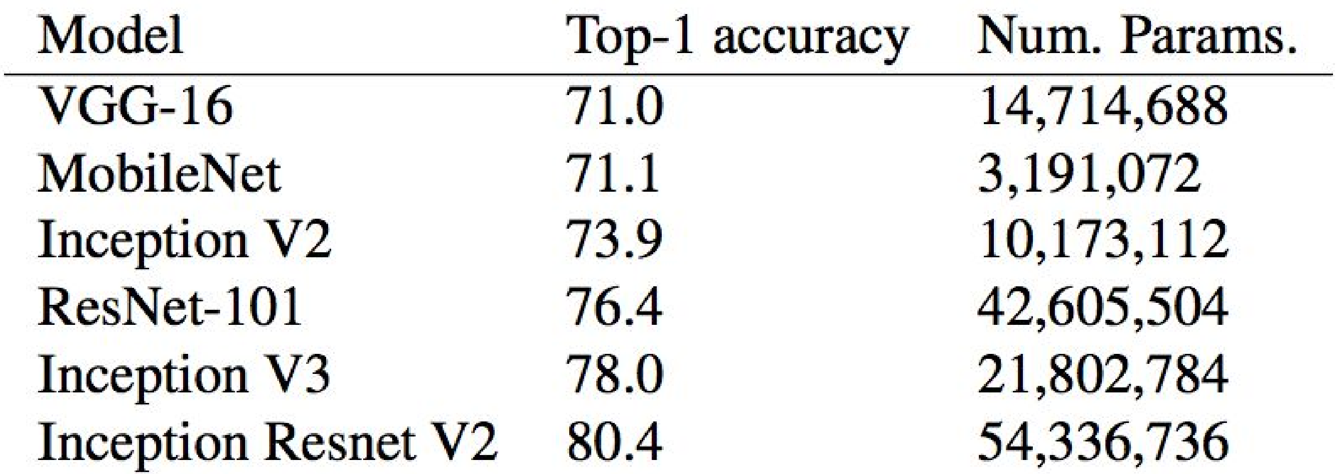
Object Detection Meta-Architecture Type 3: Region-Based Fully Convolutional (R-FCN)



Google

[Dai et al., 2015]

Wide Choice of Feature Extractors
Accuracy on ImageNet vs. model size



Model	Top-1 accuracy	Num. Params.
VGG-16	71.0	14,714,688
MobileNet	71.1	3,191,072
Inception V2	73.9	10,173,112
ResNet-101	76.4	42,605,504
Inception V3	78.0	21,802,784
Inception Resnet V2	80.4	54,336,736

Build Your Own Object Detector -- Lots of Combinations!

Meta Architecture

1. SSD
2. Faster R-CNN
3. R-FCN

×

Feature Extractor

1. Inception Resnet V2
2. Inception V2
3. Inception V3
4. MobileNet
5. Resnet 101
6. VGG 16

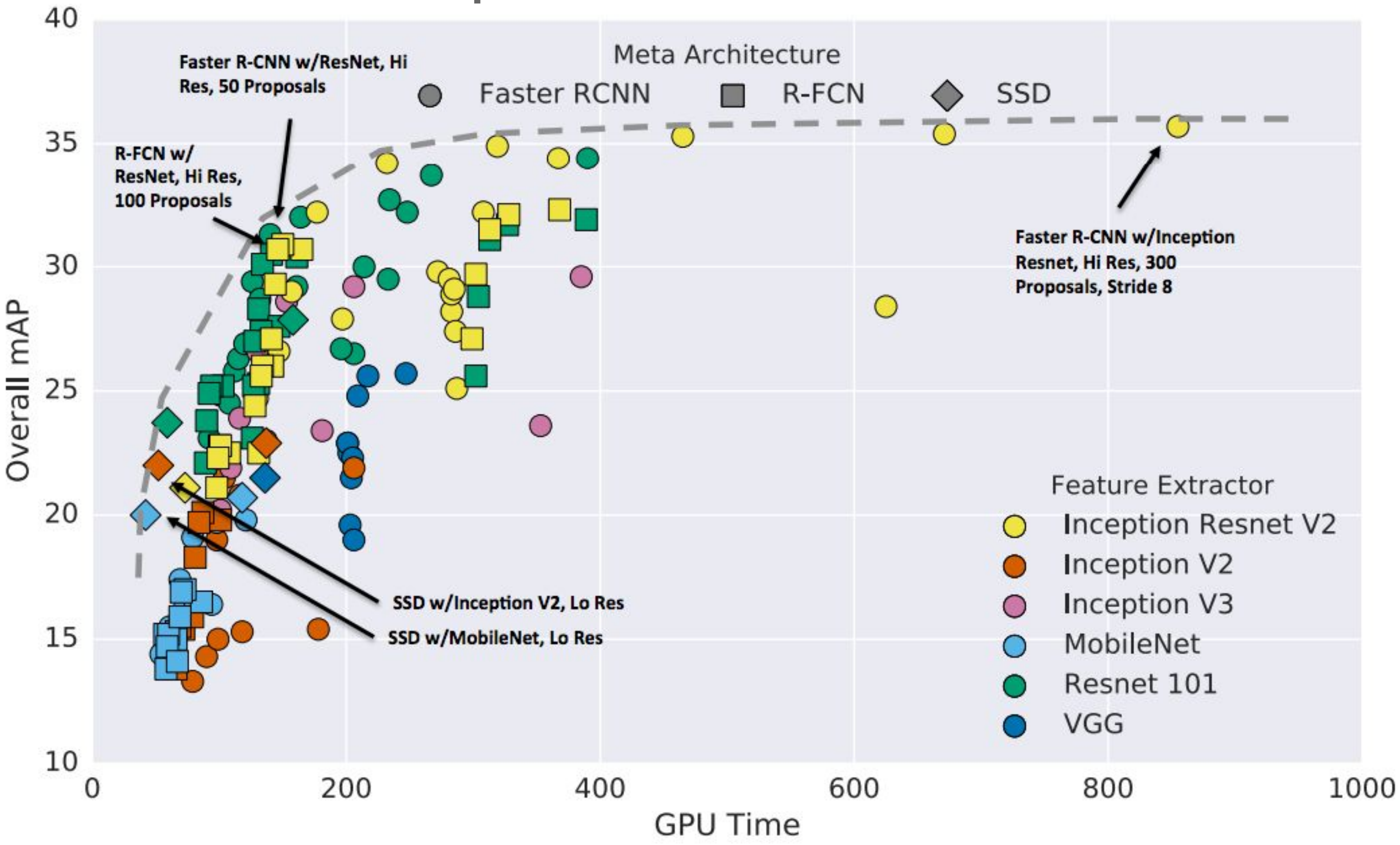
×

Other Important Choices

- Input: low-res, hi-res
- Match: argmax, bipartite,...
- Location loss: smooth L1,
- Bounding box encoding
- Stride
- # Proposals
- Other hyperparameters...

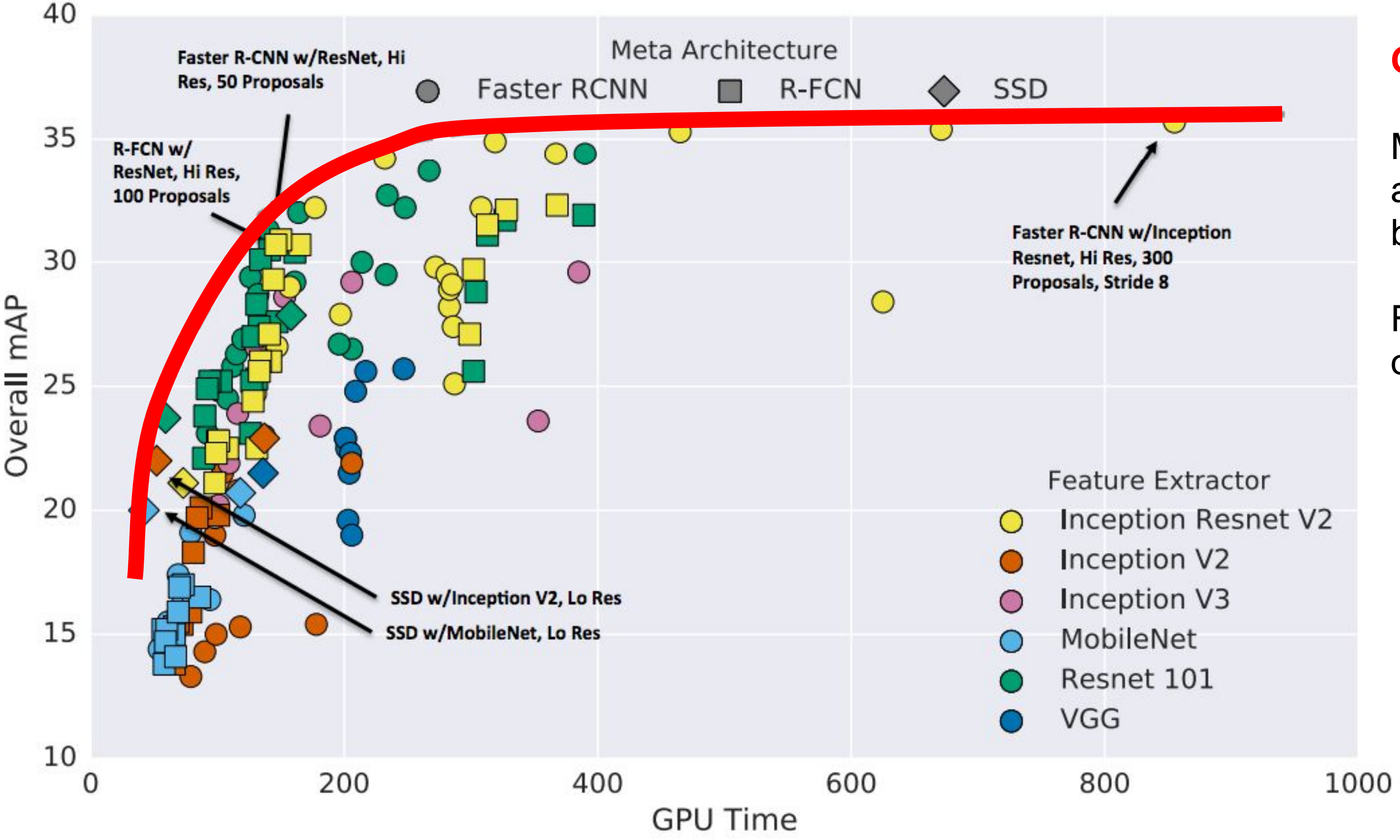
[Huang et al.] evaluate ~150 combinations in the paper!

mAP vs. Computation



Google

mAP vs. Computation



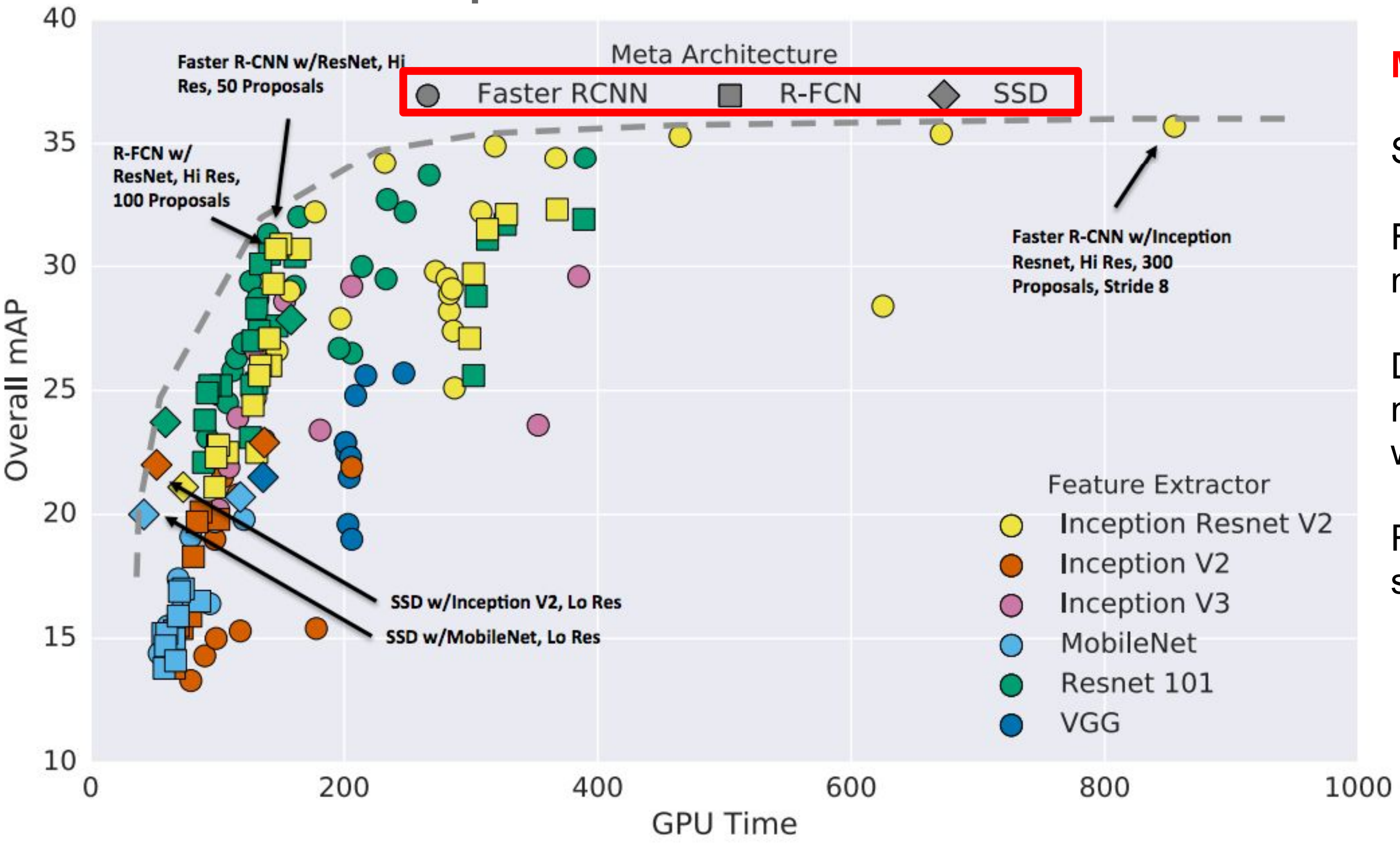
Optimality “Frontier”

Models below the curve are generally dominated, both in accuracy & speed

Focus discussion on the ones close to the curve

Google

mAP vs. Computation



Meta architecture

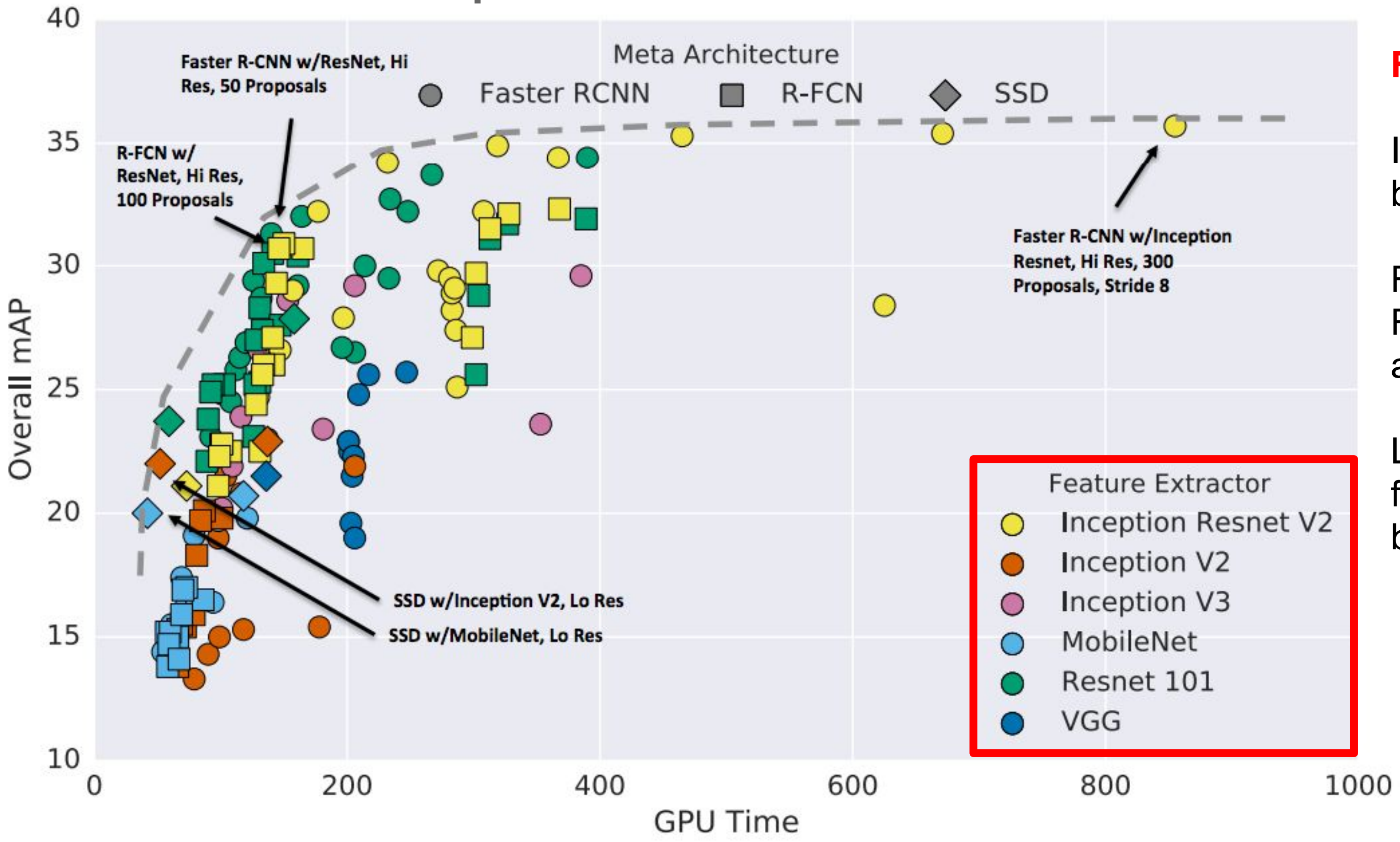
SSD models are fastest

Faster R-CNN is slow but more accurate

Dropping #proposals makes Faster R-CNN fast w/o much mAP drop

R-FCN is close to that sweet spot

mAP vs. Computation



Feature Extractor

Inception Resnet V2 gives best mAP

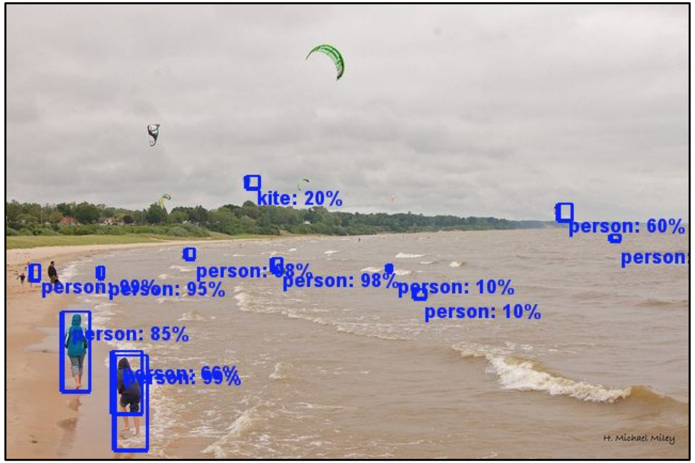
ResNet (either with R-FCN or Faster R-CNN) are is at the “elbow”

Low-res: Mobilenet is fastest but Inception V2 is bit better (& slower)

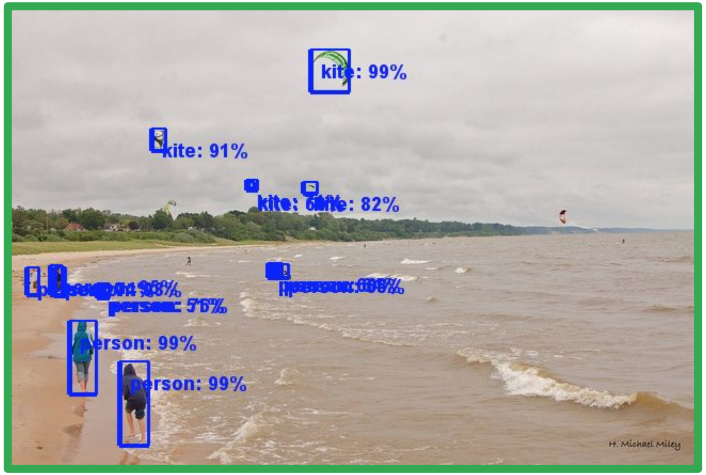
Google

Qualitative Comparison

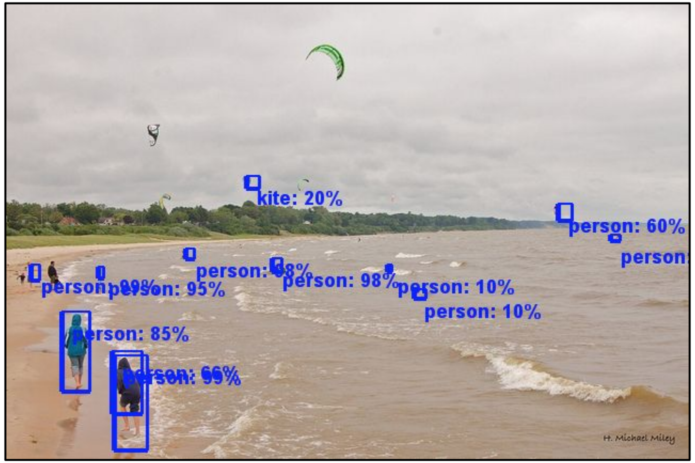
Inception Resnet SSD



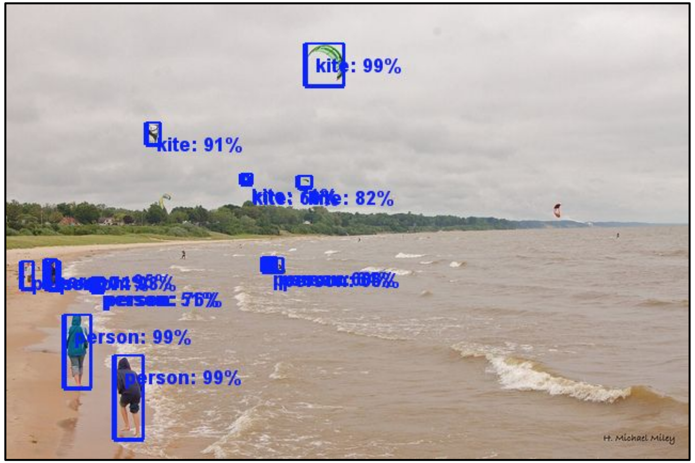
Resnet Faster RCNN



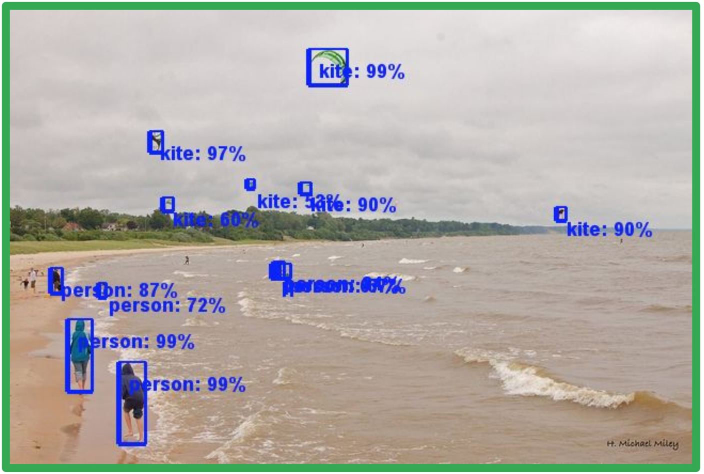
Inception Resnet SSD



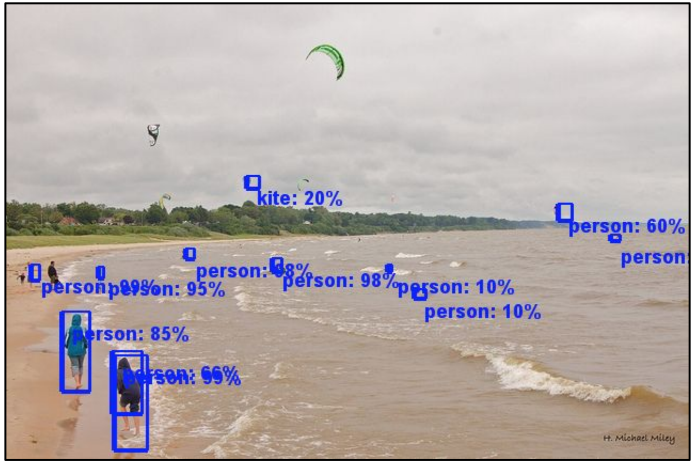
Resnet Faster RCNN



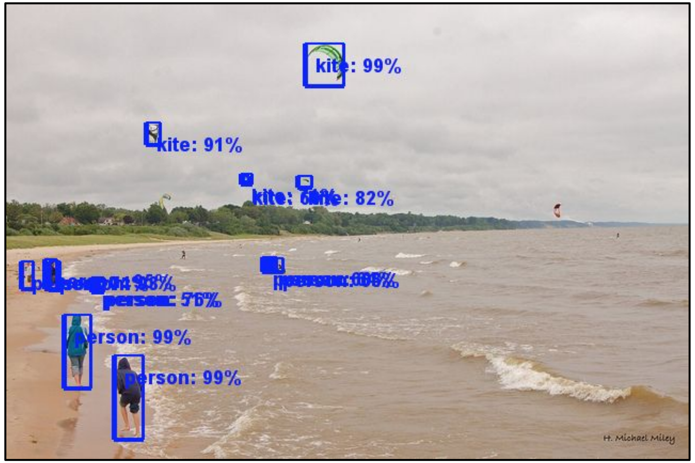
Inception Resnet Faster RCNN



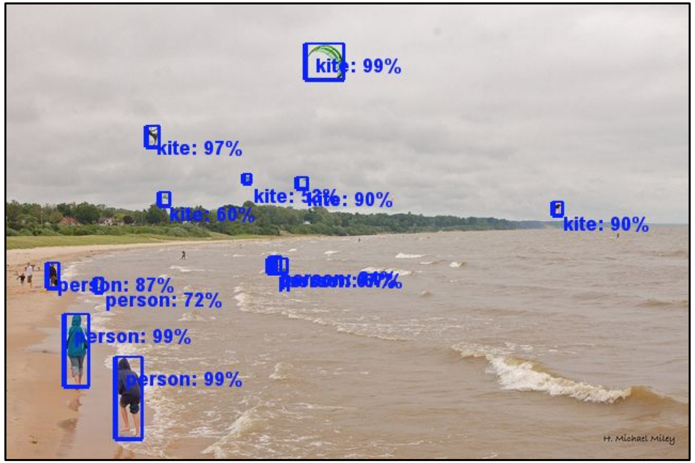
Inception Resnet SSD



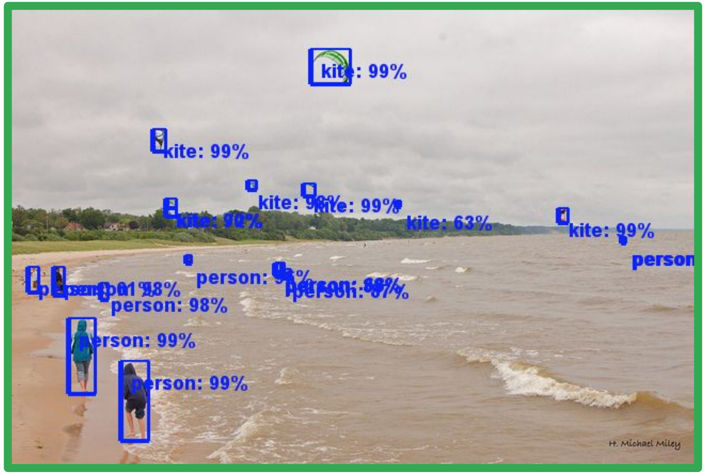
Resnet Faster RCNN



Inception Resnet Faster RCNN



Final ensemble with multicrop inference

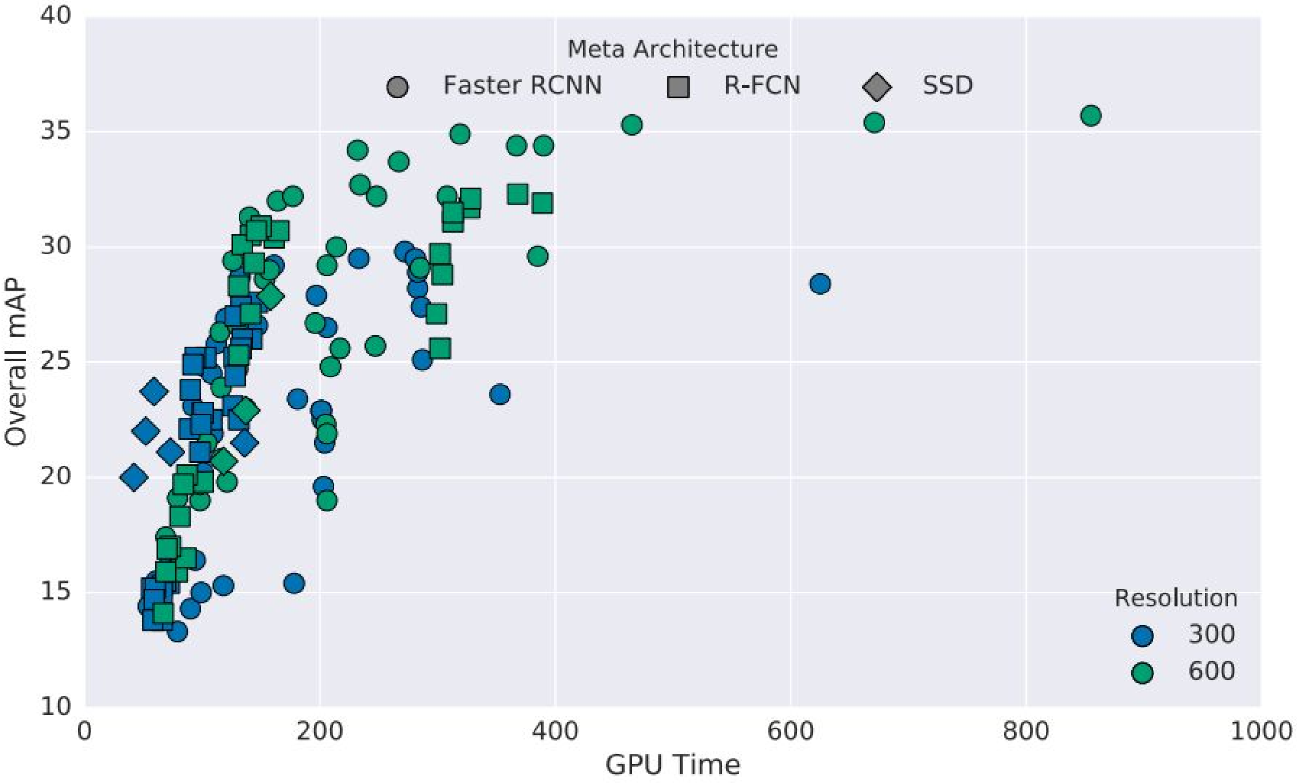


Object Detection mAP -- Input Image Size

Lo-res is 27% more efficient but 16% less mAP.

Hi-res much better for small objects.

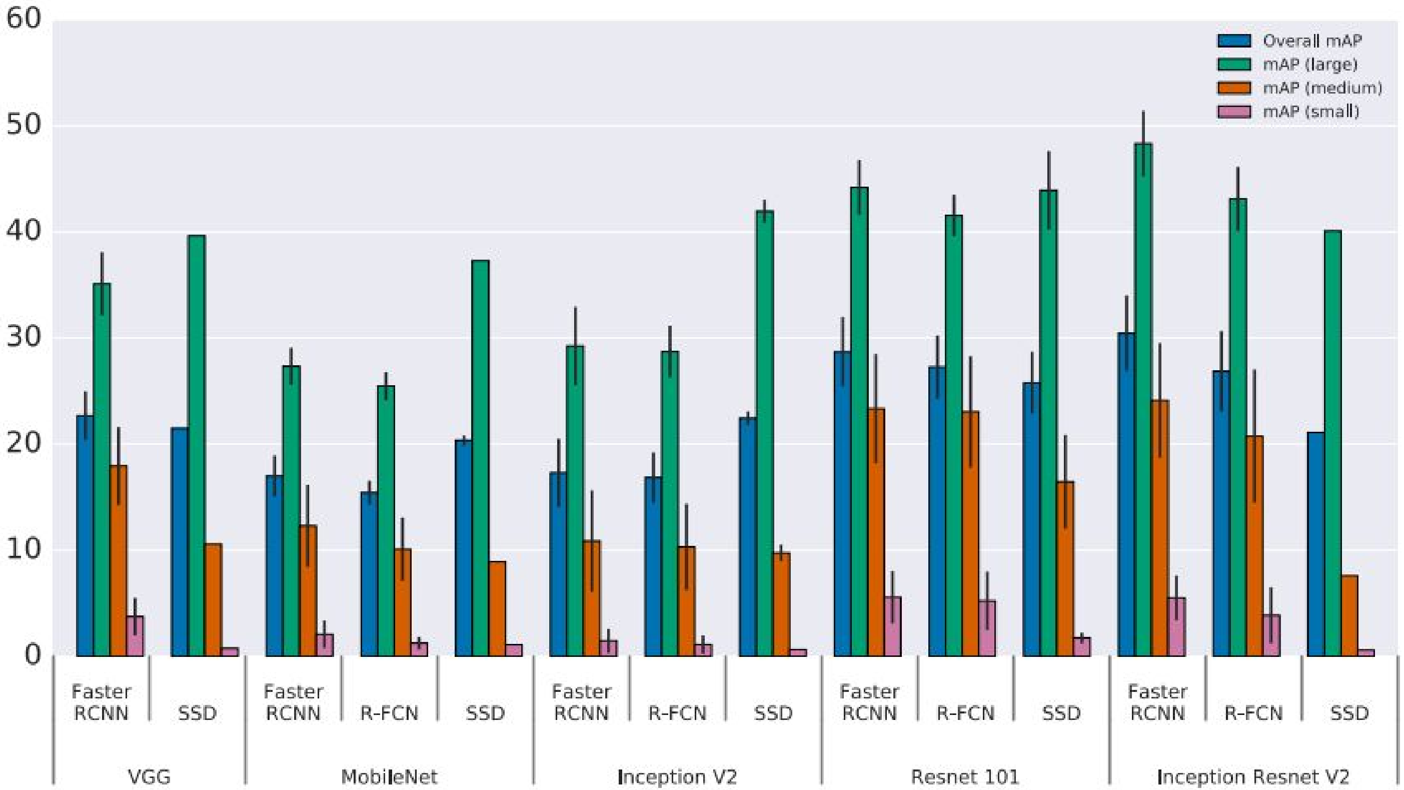
Models that do well on small objects also do well on large (but converse is not true)



Google

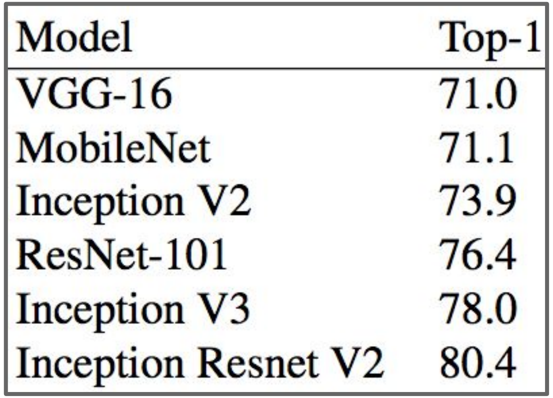
Object Detection mAP -- Small vs. Large Objects

- Large objects are easier to detect (for all models)
- SSD is particularly bad on small objects but good on larger

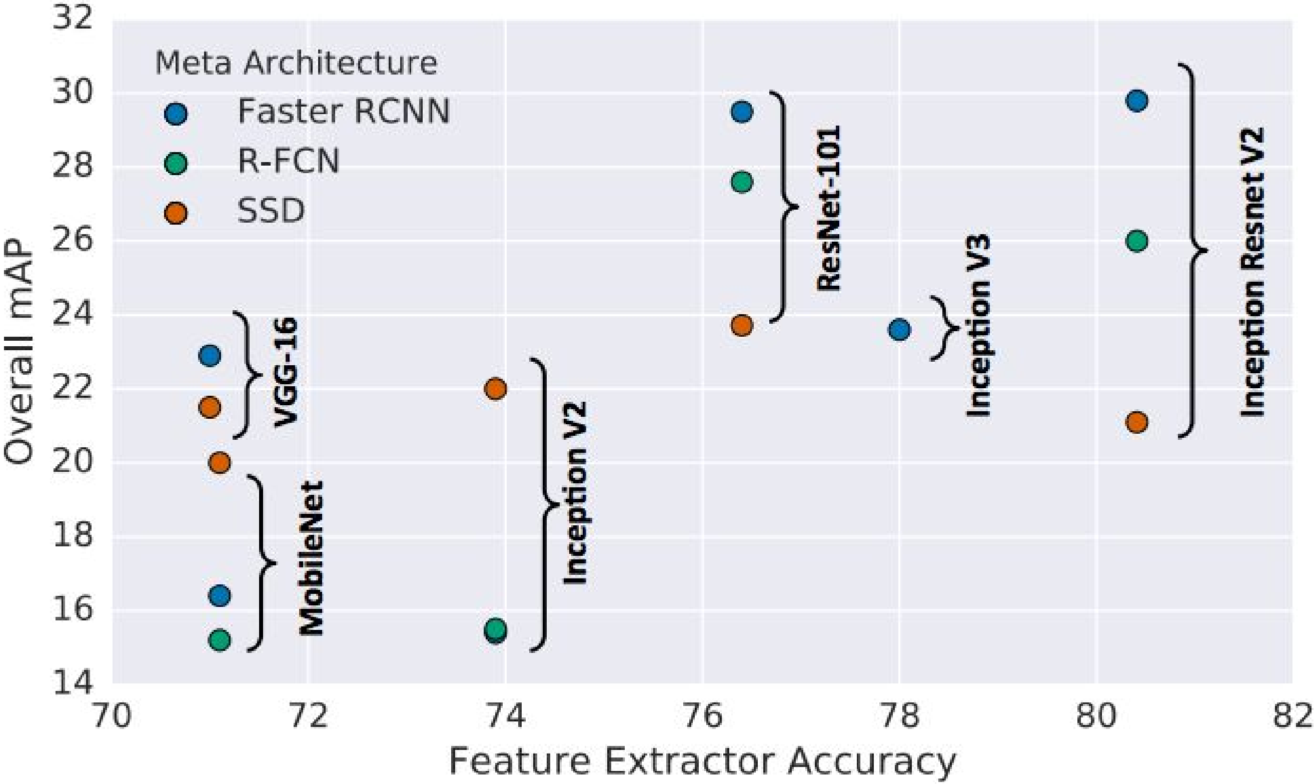


Google

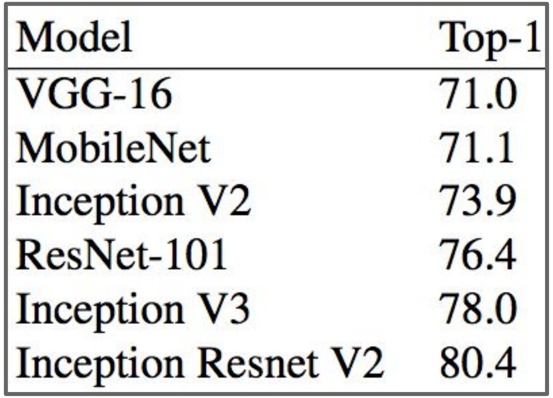
Object detector performance (mAP on COCO) vs. feature extraction accuracy (top-1% on ImageNet)



Model	Top-1
VGG-16	71.0
MobileNet	71.1
Inception V2	73.9
ResNet-101	76.4
Inception V3	78.0
Inception Resnet V2	80.4

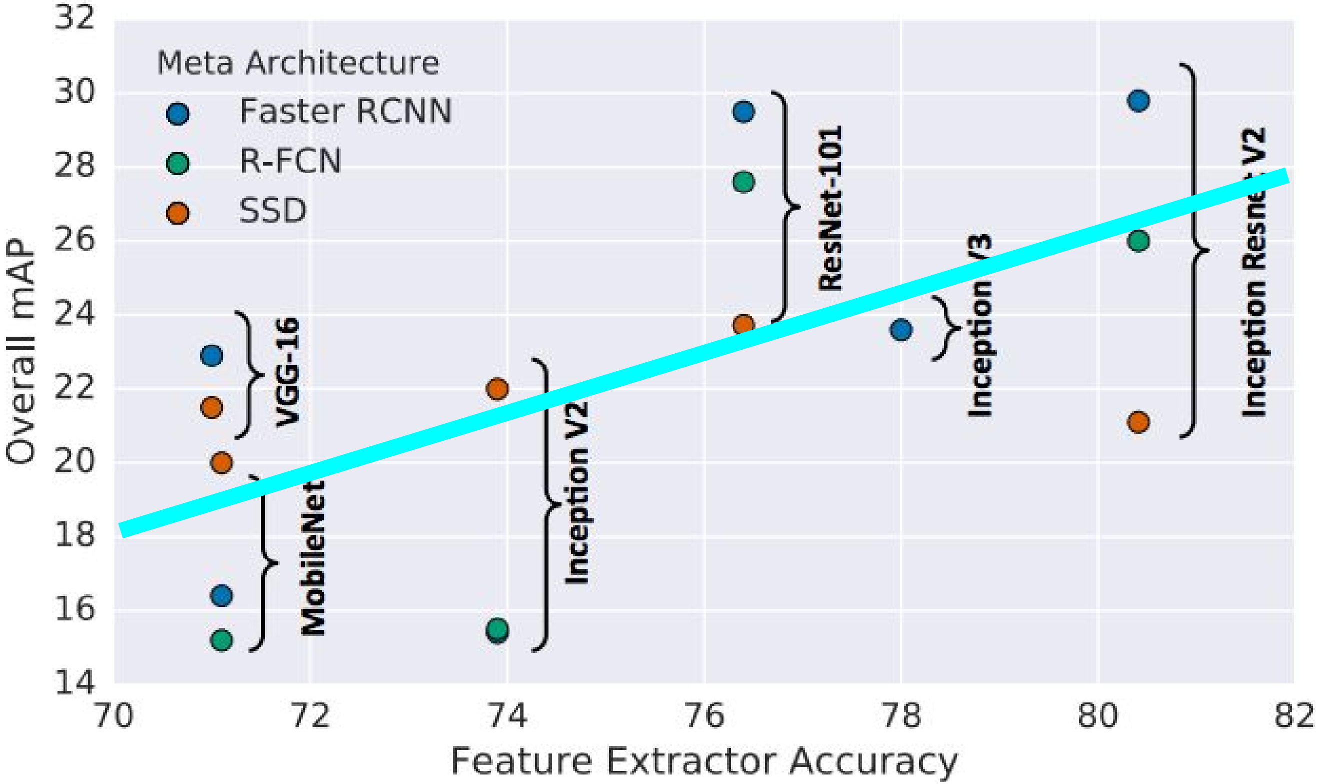


Object detector performance (mAP on COCO) vs. feature extraction accuracy (top-1% on ImageNet)

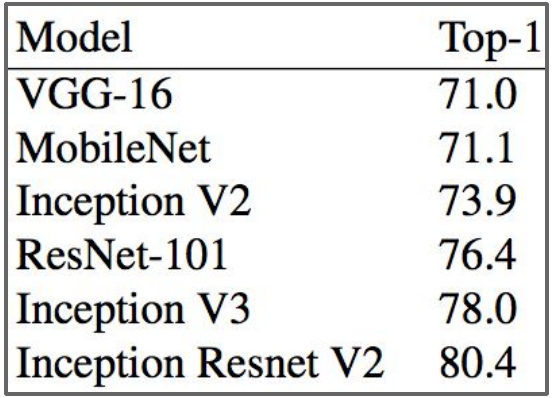


Model	Top-1
VGG-16	71.0
MobileNet	71.1
Inception V2	73.9
ResNet-101	76.4
Inception V3	78.0
Inception Resnet V2	80.4

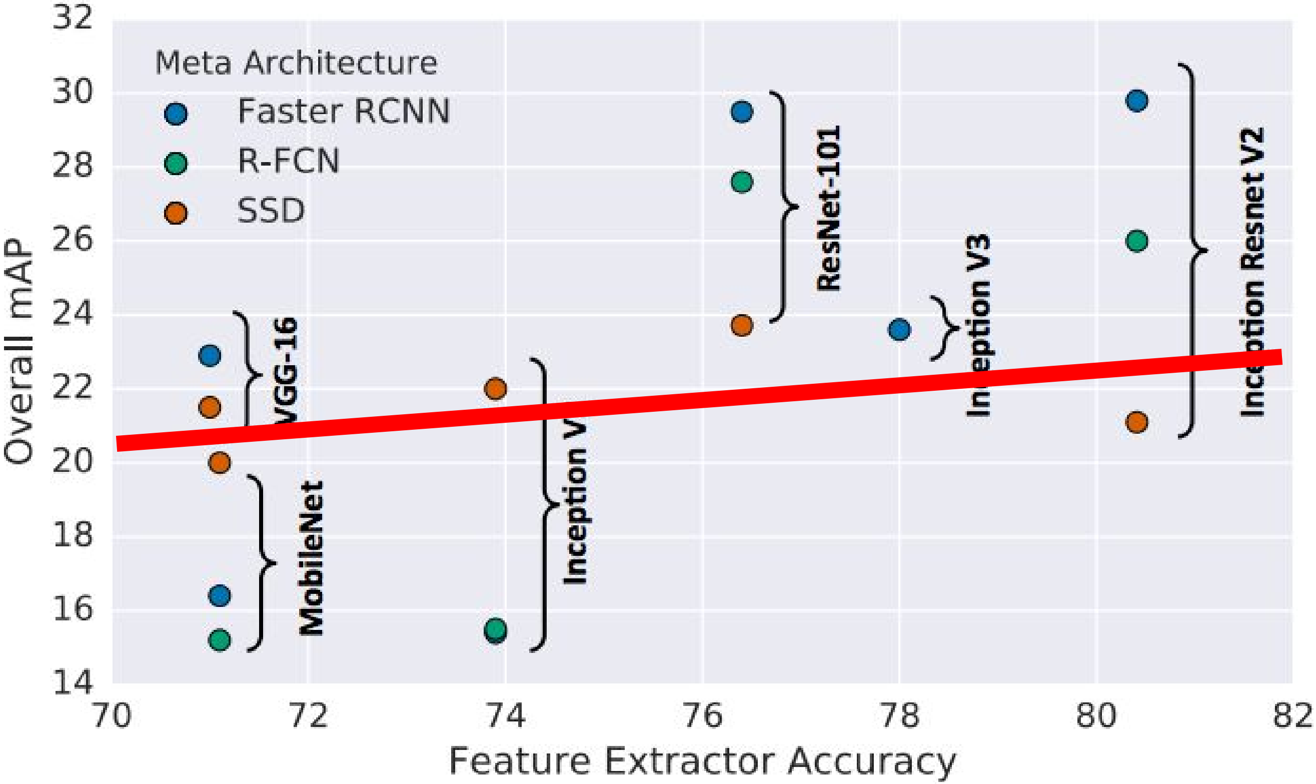
- Overall correlation -- better feature extractor does help



Object detector performance (mAP on COCO) vs. feature extraction accuracy (top-1% on ImageNet)



Model	Top-1
VGG-16	71.0
MobileNet	71.1
Inception V2	73.9
ResNet-101	76.4
Inception V3	78.0
Inception Resnet V2	80.4



- Overall correlation -- better feature extractor does help
- But not as much for SSD...

Summary of Part 1 (Object Detection)

Object detection has had a revolution in just the last few years!

- Hand-crafted features ⇒ Deep-learned features
- Sliding windows ⇒ Segmentation-based proposals ⇒ Deep-learned proposals
- Part-based models ⇒ Linear SVMs (R-CNN) ⇒ No more SVMs :-)

Currently in a very empirical phase, where intuitions cannot guide us reliably.

That’s why I think that work like [Huang et al., 2016] is valuable...

Part 2: Image Compression (with neural nets, of course!)

Toderici, O'Malley, Hwang, Vincent, Minnen, Baluja, Covell, Sukthankar
“Variable Rate Image Compression with Recurrent Neural Networks”
https://arxiv.org/abs/1511.06085

Toderici, Vincent, Johnston, Hwang, Minnen, Shor, Covell
“Full Resolution Image Compression with Recurrent Neural Networks”
https://arxiv.org/abs/1608.05148

Neural Net Based Image Compression

- Confluence of two research areas:
  - “Classical” image compression (e.g., JPG, WebP, BPG)
  - Neural networks -- particularly the idea auto-encoders

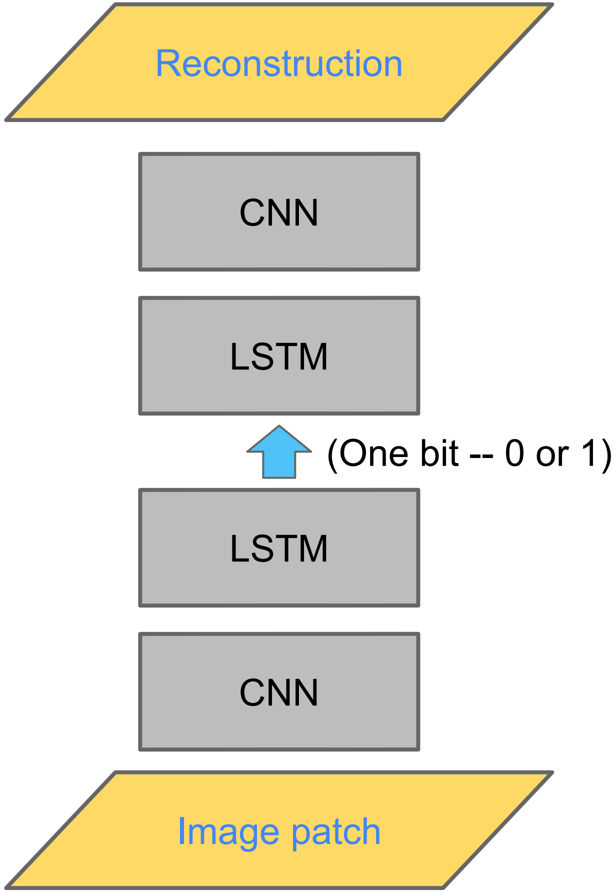
- Devil is in the details
  - How do we know it’s doing a good job?
  - Proposed application imposes requirements
    - Compression for transmission vs. back-end storage
    - Thumbnails vs. full-sized images vs. video

Why expect this to improve systems engineered by 1000s of people over 10+ years?

Design Challenges

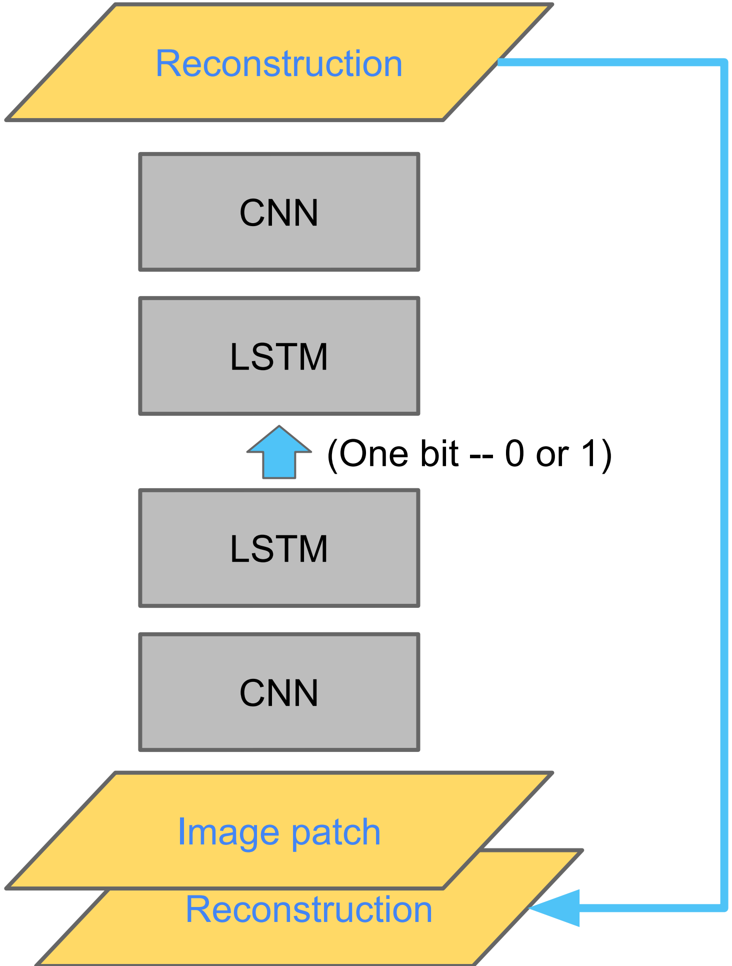
- Lossy compression OK but need high quality output from a *human* perspective
- Competitive compression by learning from lots of (unsupervised) data
    - Selecting the right data is key! (cf. hard negative mining)
- Binarization: generating *bits* from neural nets and doing end-to-end learning
- Variable bitrate: tuning compression ratio without retraining the model
- Adaptive bitrate: allocating more bits to more complex regions

Initial (Raw) Idea

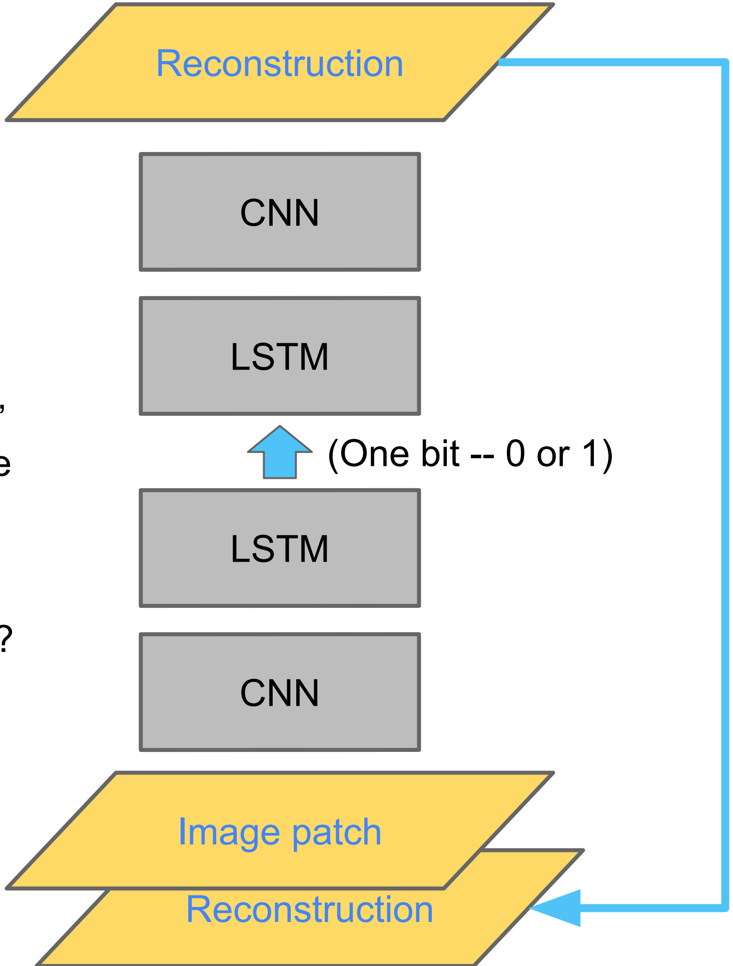


Google

Initial (Raw) Idea



Initial (Raw) Idea



Benefits

- LSTMs invent a “language”
- Progressive -- stop anytime

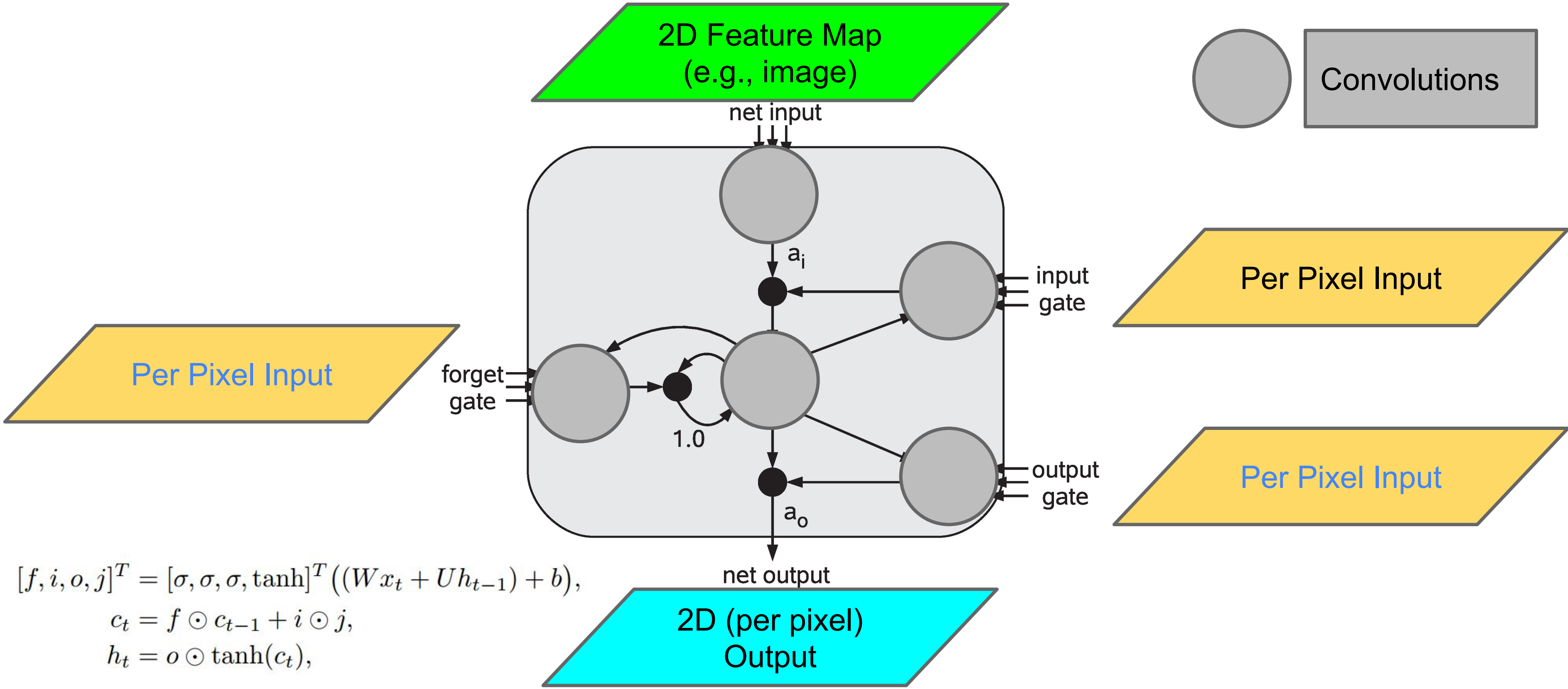
Issues to consider

- Greedy but not optimal?
- Learning through binarizer?
- Just one bit at a time?
- Is this the right input?
- Is this the right output?
- Variable size images?

Choice of Recurrent Units (RNN)

- Allows a neural network to keep “state” (enables sequence processing)
- Many types of RNN:
    - Vanilla RNN -- suffers from vanishing gradients
    - Long Short Term Memory (LSTM) -- a variant which does not
    - Gated Recurrent Units (GRU) - another RNN that also does not
    - Associative Long Short Term Memory - newer variant of LSTM
    - Residual GRU - new proposed RNN for compression
- We’ll use LSTM as an example, but have experiments with all

Building block: Conv2D Extension of Recurrent Networks



2D Feature Map
(e.g., image)

Convolutions

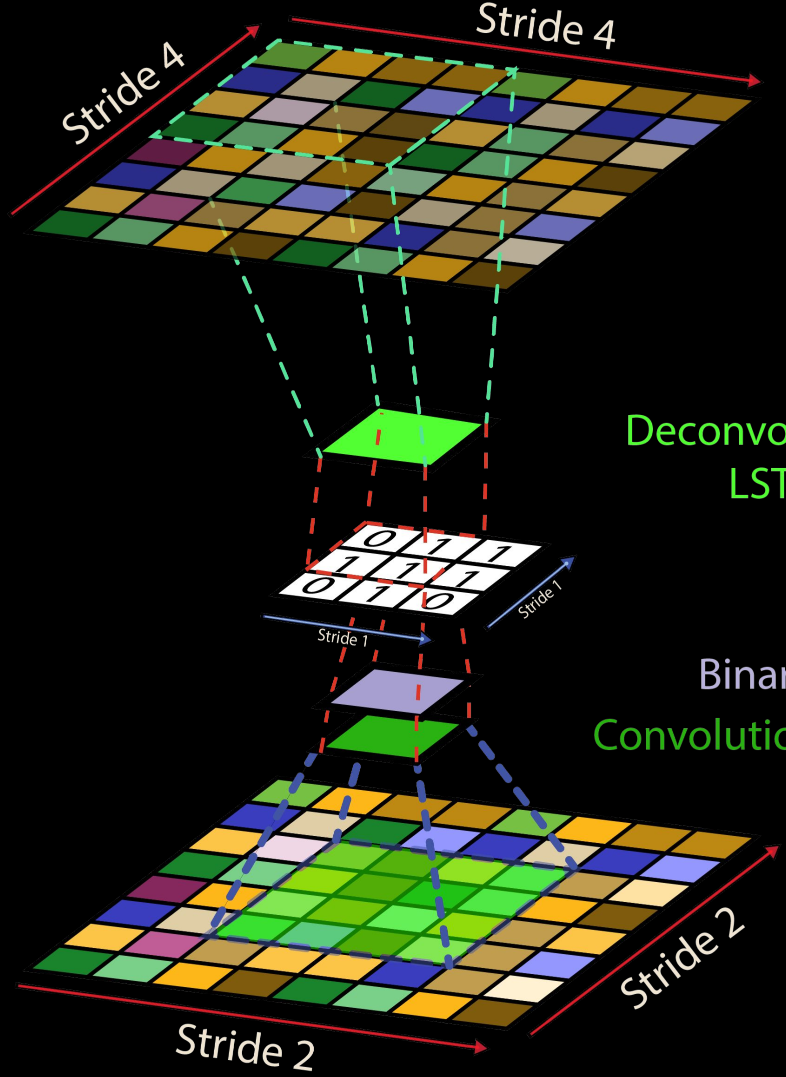
Per Pixel Input

Per Pixel Input

Per Pixel Input

2D (per pixel)
Output

$$
\begin{aligned}
[f, i, o, j]^T &= [\sigma, \sigma, \sigma, \tanh]^T \big((Wx_t + Uh_{t-1}) + b\big), \\
c_t &= f \odot c_{t-1} + i \odot j, \\
h_t &= o \odot \tanh(c_t),
\end{aligned}
$$

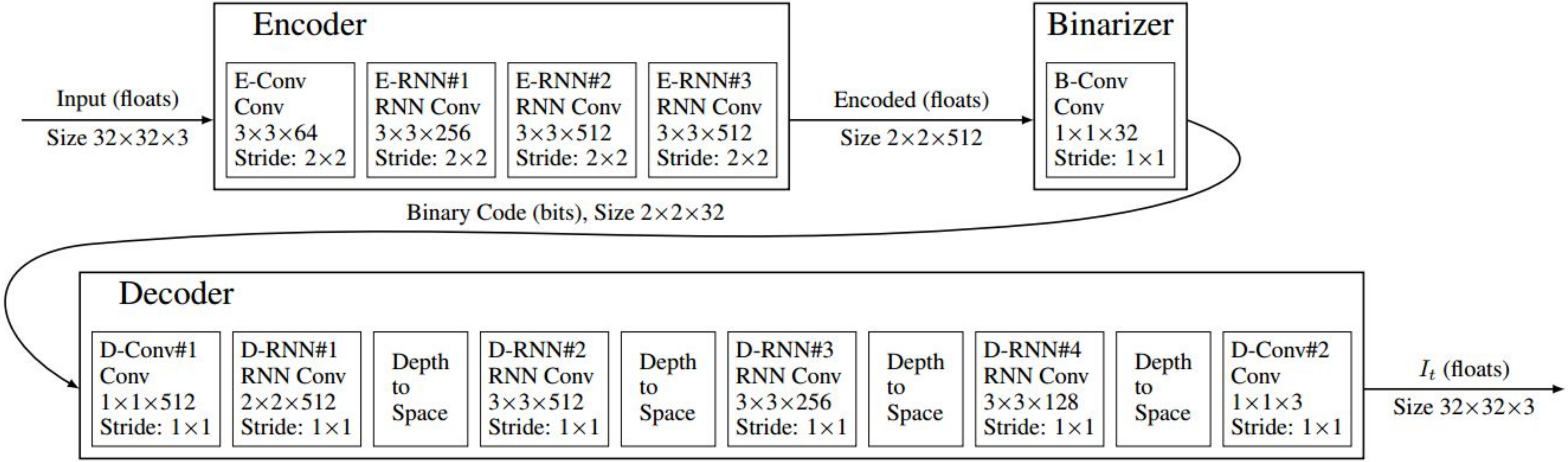


Deconvolutional LSTM

Binarizer

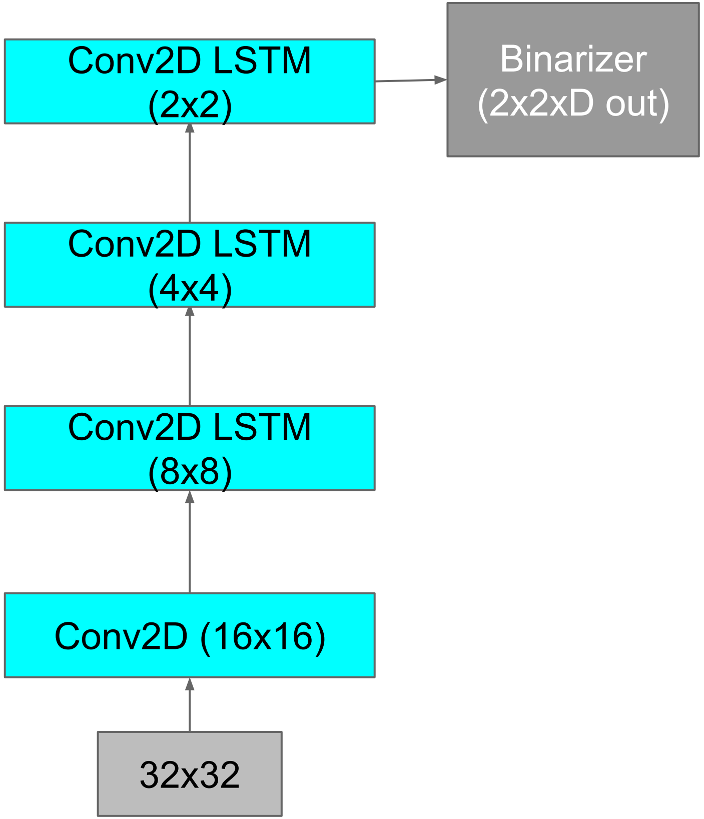
Convolutional LSTM

General RNN Architecture (1 Iteration)



Google

Encoder / Binarizer



Binarizer (2x2xD out)

Conv2D LSTM (2x2)

Conv2D LSTM (4x4)

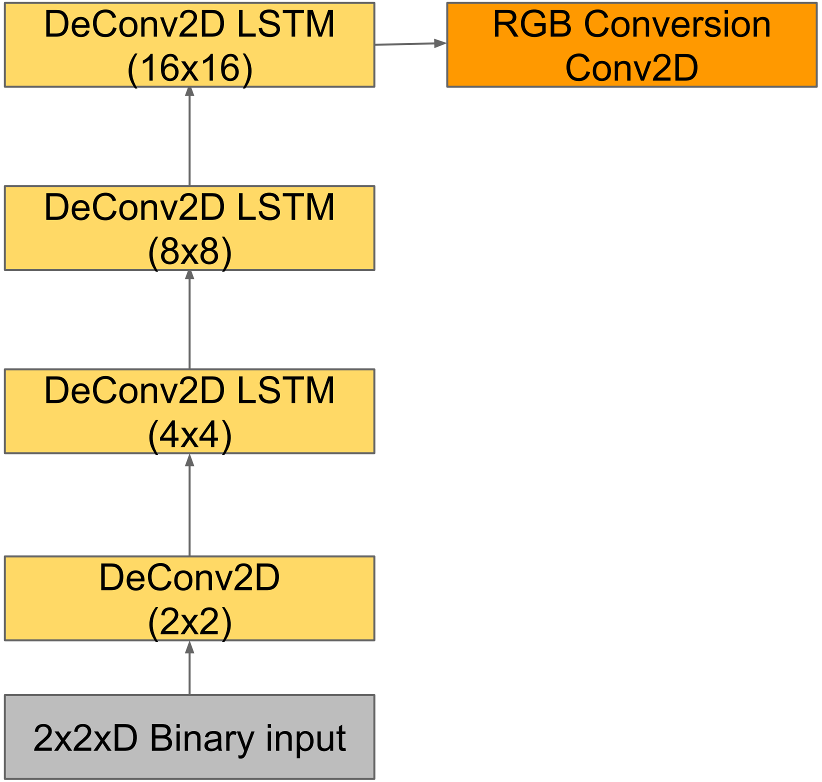
Conv2D LSTM (8x8)

Conv2D (16x16)

32x32

- The **training** binarizer consists of the following:
  - 1x1 Conv2D + tanh activation
  - The final output of the layer is:
    - b(x) = 1, iff x > u, u ~ U[-1, 1],
      -1, otherwise
- The eval/compression binarizer:
  - 1x1 Conv2D + tanh activation
  - The final output of the layer is:
    - b(x) = 1, if x > 0
      -1, otherwise
- The output from the binarizer is the pre-entropy coding data stream
  - entropy coding is not necessary unless wanting best possible compression ratio

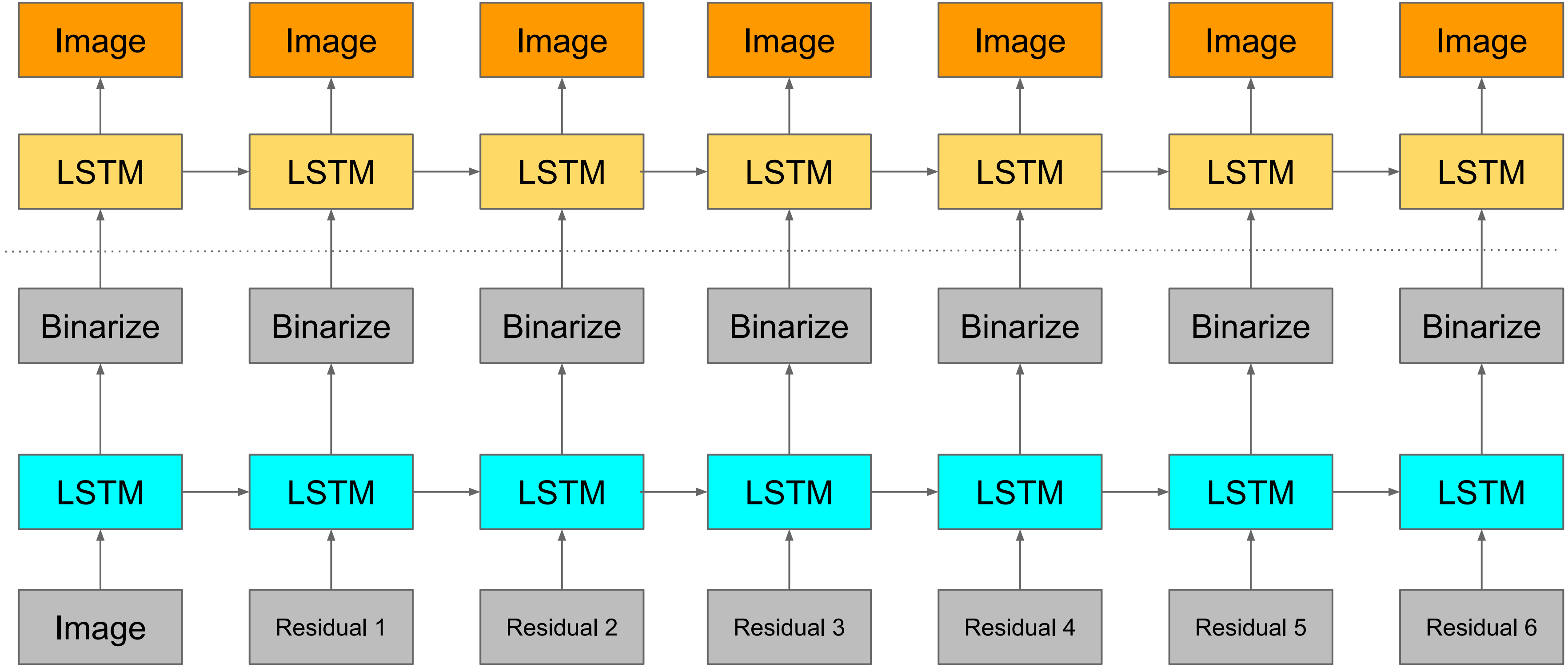
Decoder



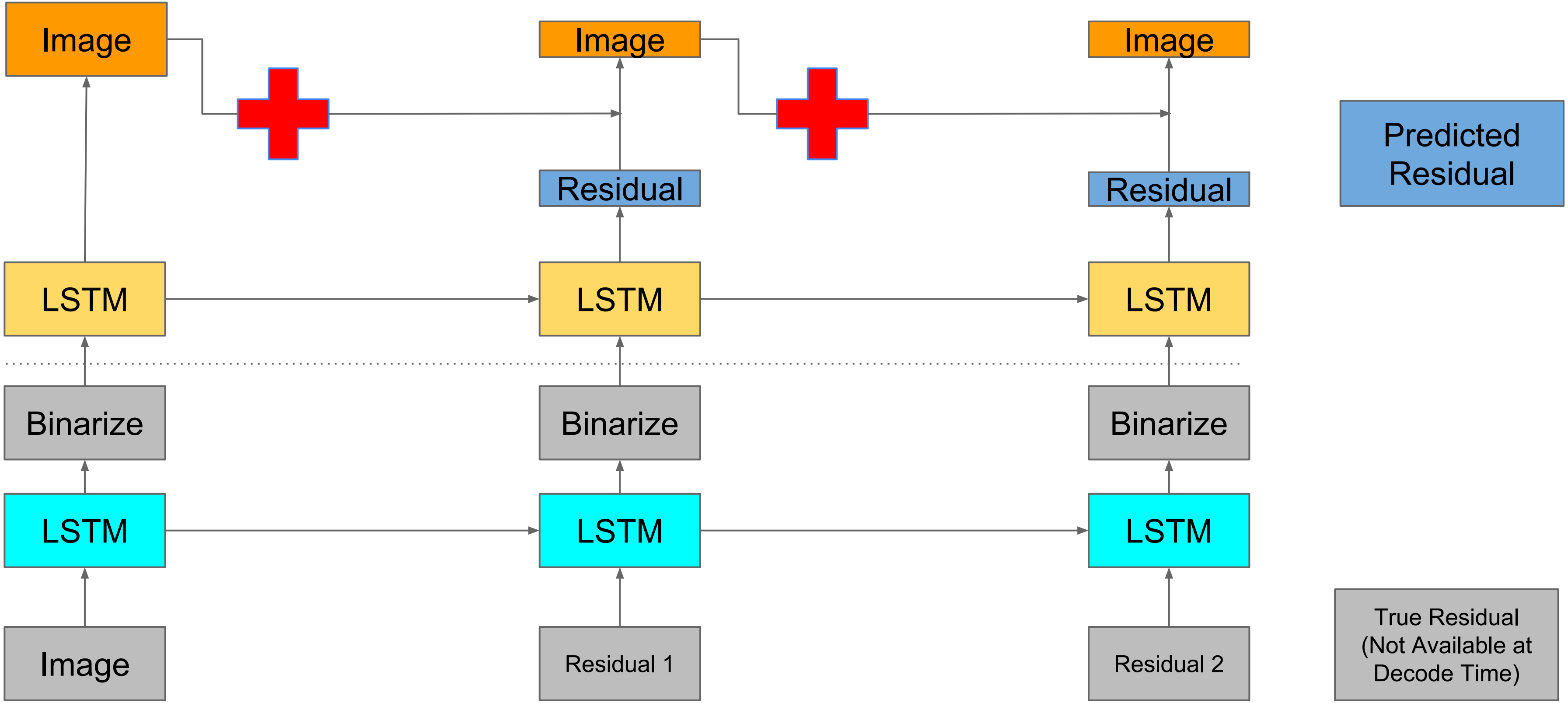
- Here we increase spatial resolution by a factor of 2 in each direction at each step.
- DeConv2DLSTM(x) = tf.depth2space(Conv2DLSTM(x), 2)*
- DeConv2D(x) = tf.depth2space(Conv2D(x), 2)

* tf.depth_to_space is a TensorFlow function

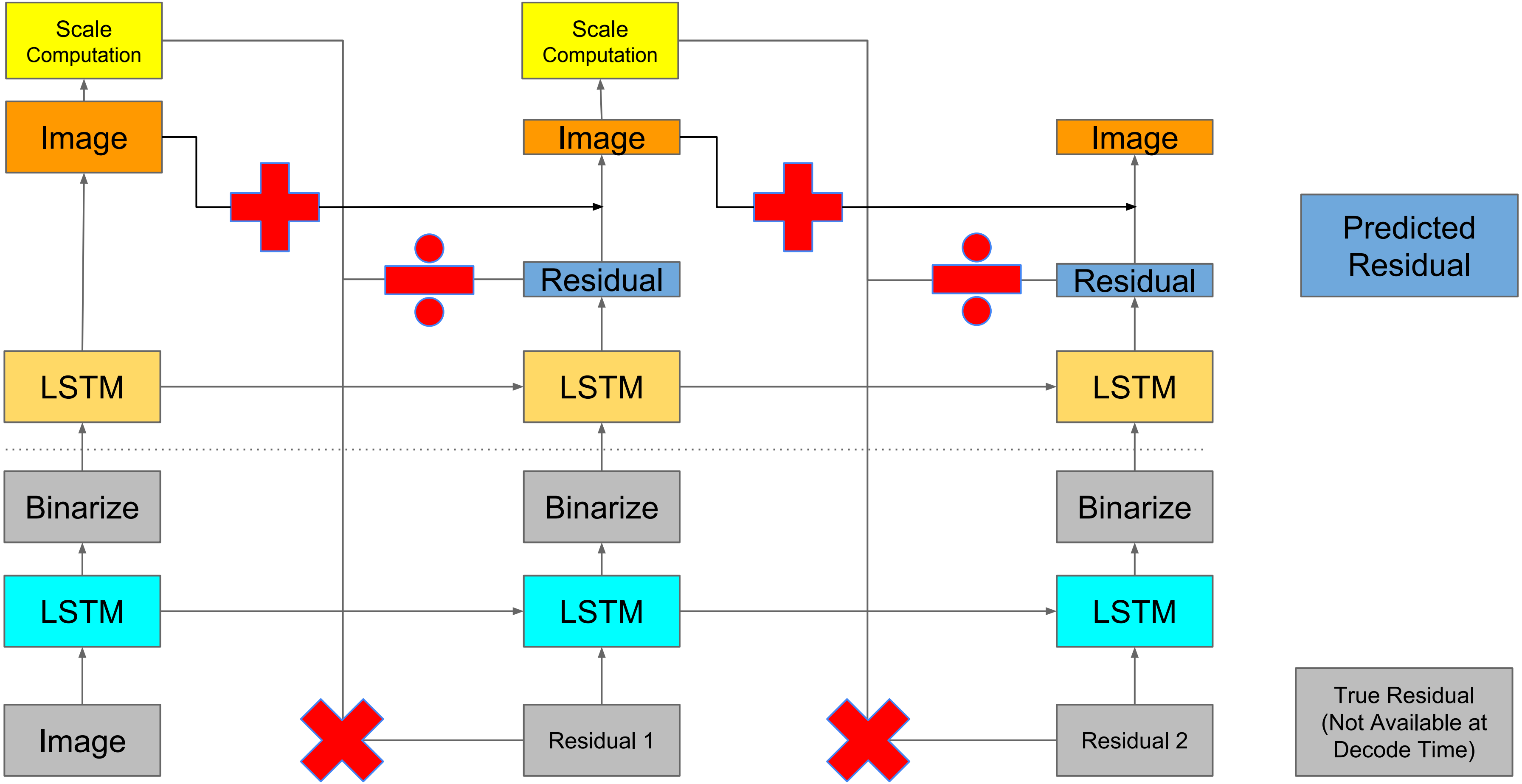
Image/Frame Compression (“One Shot”)



Image/Frame Compression ("Additive Reconstruction")

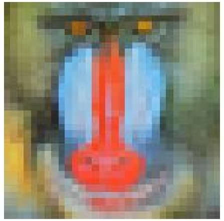


Image/Frame Compression (“Scaled Residual”)

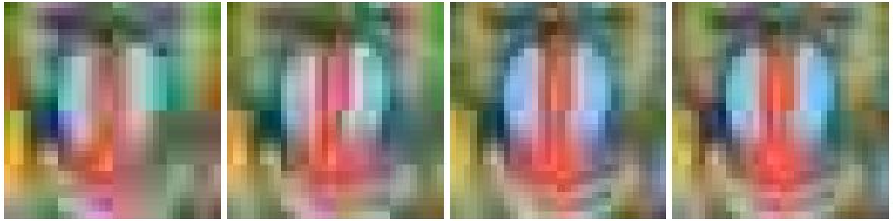


Google

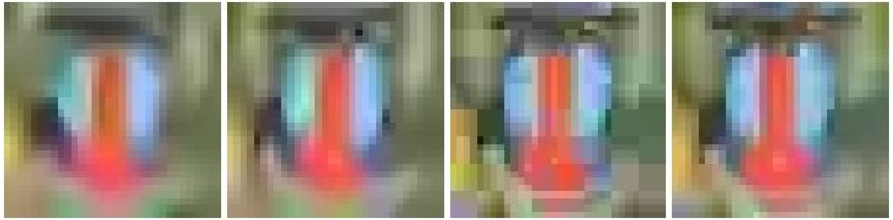
Mandrill (32x32)



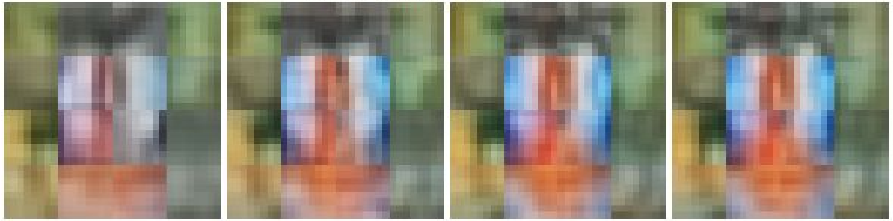
Original (32×32)



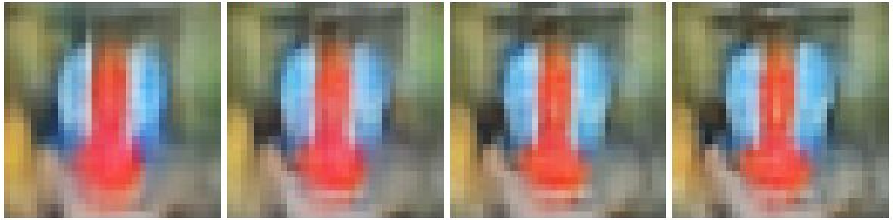
JPEG compressed images



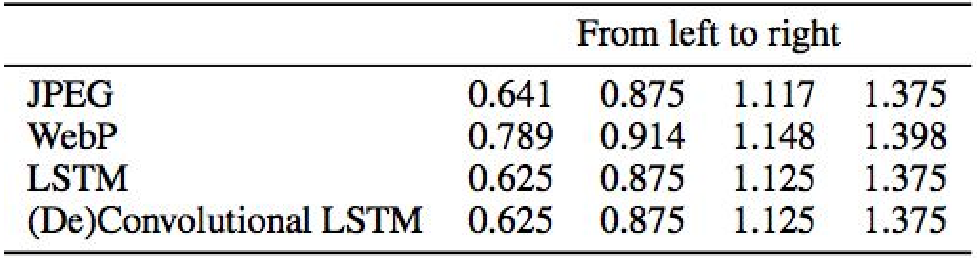
WebP compressed images



Compressed images with LSTM architecture

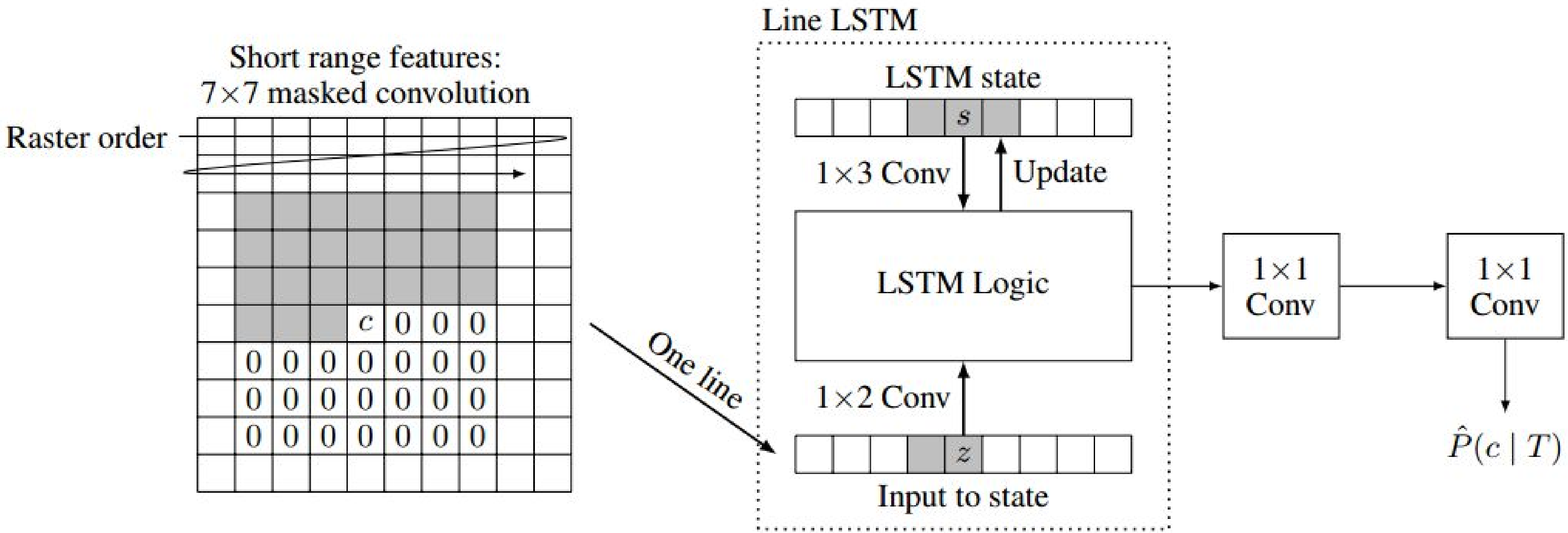


Compressed images with conv/deconv LSTM architecture



	From left to right			
JPEG	0.641	0.875	1.117	1.375
WebP	0.789	0.914	1.148	1.398
LSTM	0.625	0.875	1.125	1.375
(De)Convolutional LSTM	0.625	0.875	1.125	1.375

Entropy Coder Model (2D context + Iteration State)



Progressive Entropy Coding

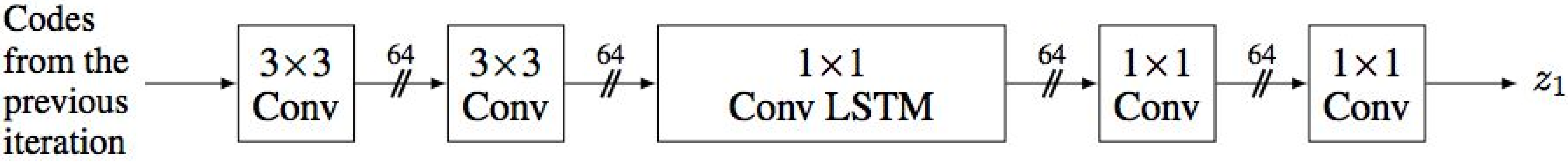


Figure 4: Description of neural network used to compute additional line LSTM inputs for progressive entropy coder. This allows propagation of information from the previous iterations to the current.

Results on Kodak (in aggregate)

- Chose MS-SSIM (RGB) and PSNR-HVS (not -M).
- They tried multiple RNNs besides LSTM (i.e., GRU, Associative Memory LSTM and a new variant of GRU).
- Best model in MS-SSIM is the worst in PSNR-HVS?!

Current best model:
1.8330 RGB MS-SSIM
vs. 1.8280 for WebP

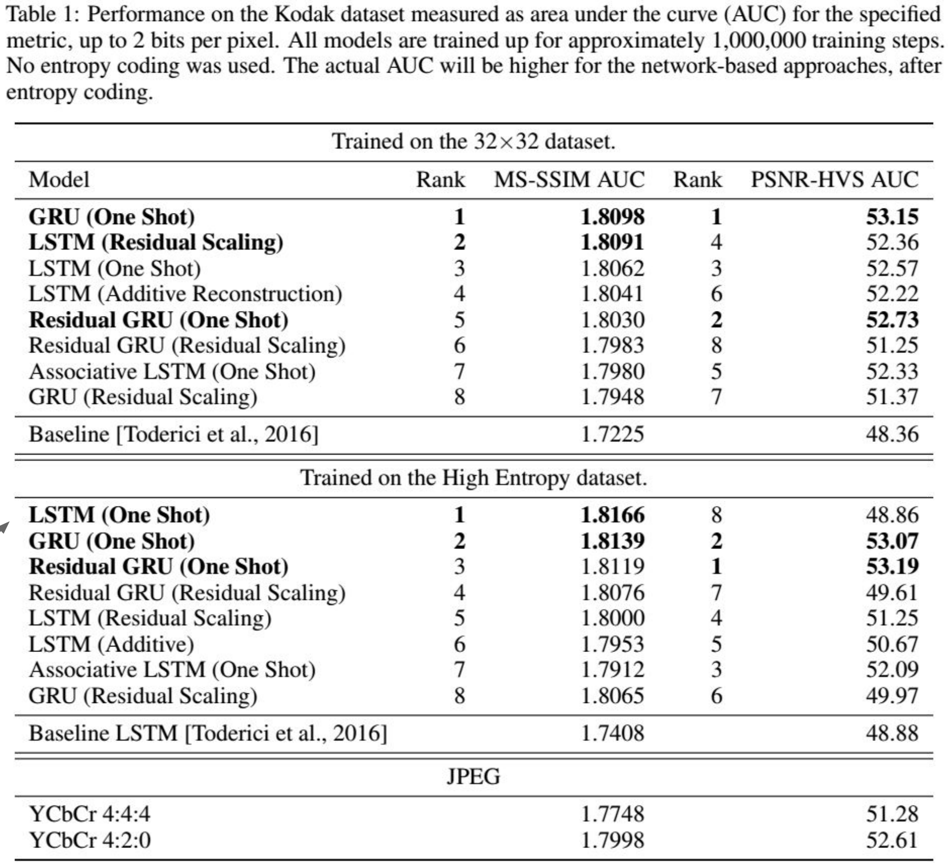
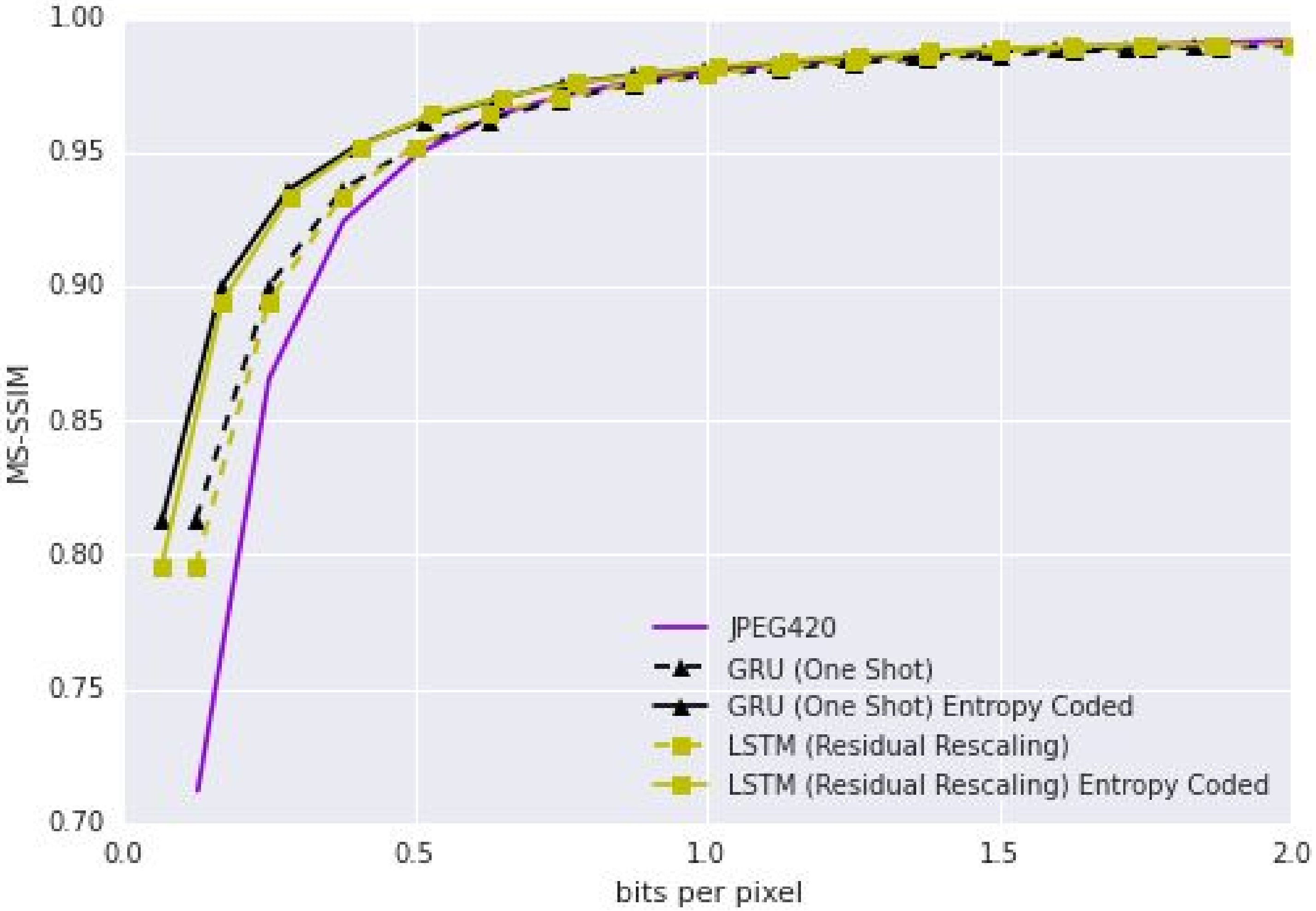


Table 1: Performance on the Kodak dataset measured as area under the curve (AUC) for the specified metric, up to 2 bits per pixel. All models are trained up for approximately 1,000,000 training steps. No entropy coding was used. The actual AUC will be higher for the network-based approaches, after entropy coding.

Model	Rank	MS-SSIM AUC	Rank	PSNR-HVS AUC
Trained on the 32×32 dataset.				
GRU (One Shot)	**1**	**1.8098**	**1**	**53.15**
LSTM (Residual Scaling)	**2**	**1.8091**	4	52.36
LSTM (One Shot)	3	1.8062	3	52.57
LSTM (Additive Reconstruction)	4	1.8041	6	52.22
Residual GRU (One Shot)	5	1.8030	**2**	**52.73**
Residual GRU (Residual Scaling)	6	1.7983	8	51.25
Associative LSTM (One Shot)	7	1.7980	5	52.33
GRU (Residual Scaling)	8	1.7948	7	51.37
Baseline [Toderici et al., 2016]		1.7225		48.36
Trained on the High Entropy dataset.				
LSTM (One Shot)	**1**	**1.8166**	8	48.86
GRU (One Shot)	**2**	**1.8139**	**2**	**53.07**
Residual GRU (One Shot)	3	1.8119	**1**	**53.19**
Residual GRU (Residual Scaling)	4	1.8076	7	49.61
LSTM (Residual Scaling)	5	1.8000	4	51.25
LSTM (Additive)	6	1.7953	5	50.67
Associative LSTM (One Shot)	7	1.7912	3	52.09
GRU (Residual Scaling)	8	1.8065	6	49.97
Baseline LSTM [Toderici et al., 2016]		1.7408		48.88
JPEG				
YCbCr 4:4:4		1.7748		51.28
YCbCr 4:2:0		1.7998		52.61

Kodak RD Curve (MS-SSIM)

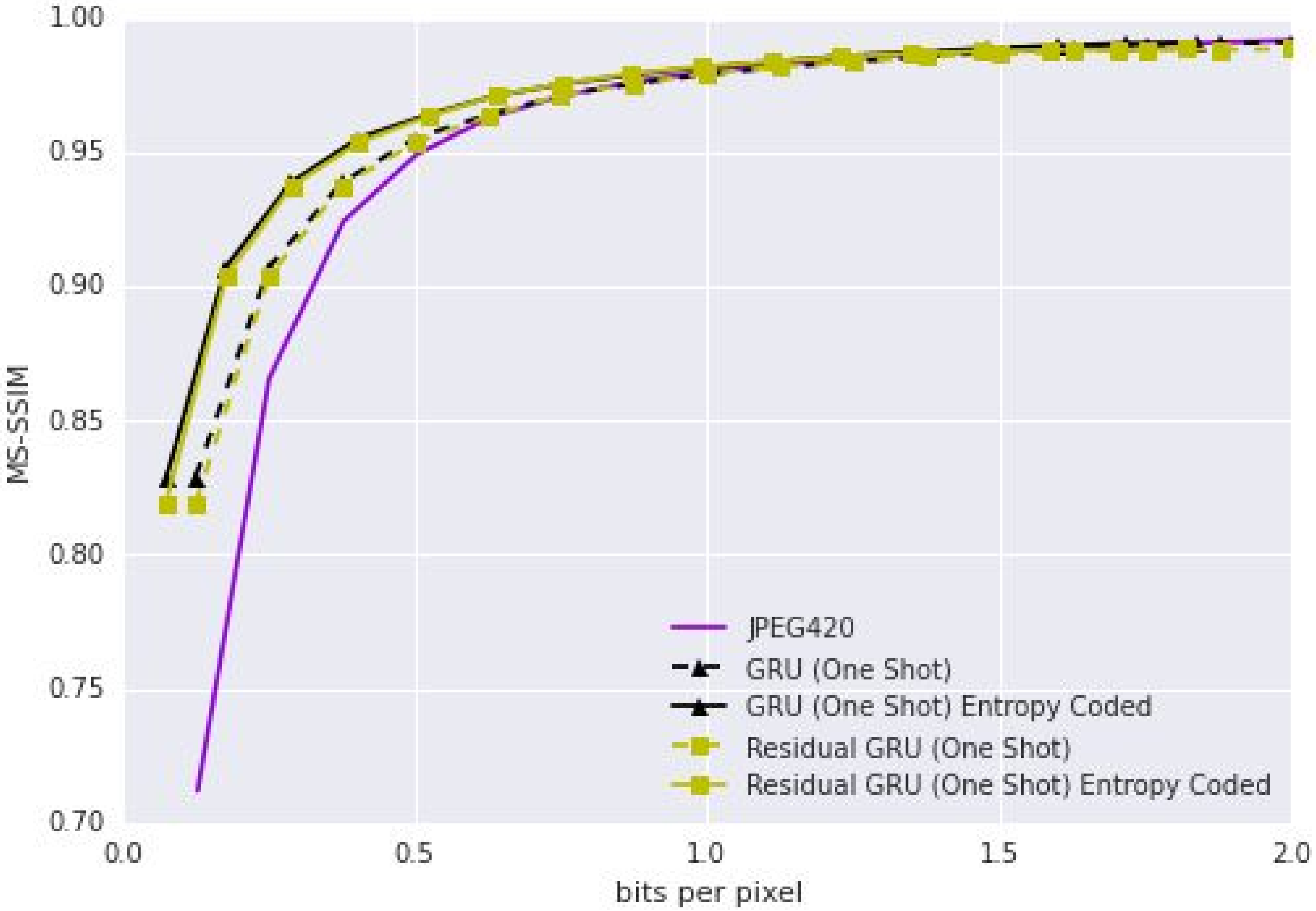
Trained on Resized 32x32 Images



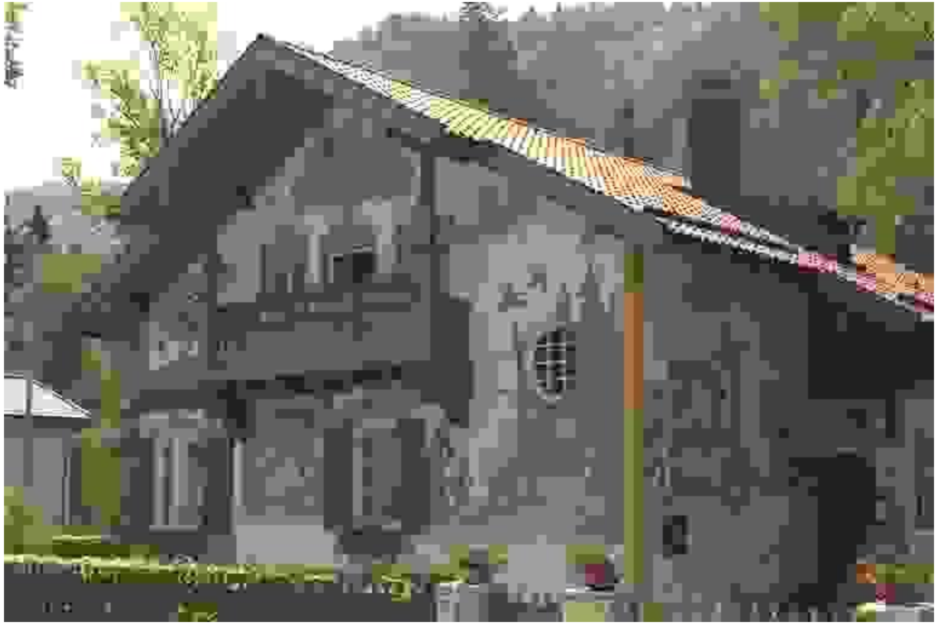
Google

Kodak RD Curve (MS-SSIM)

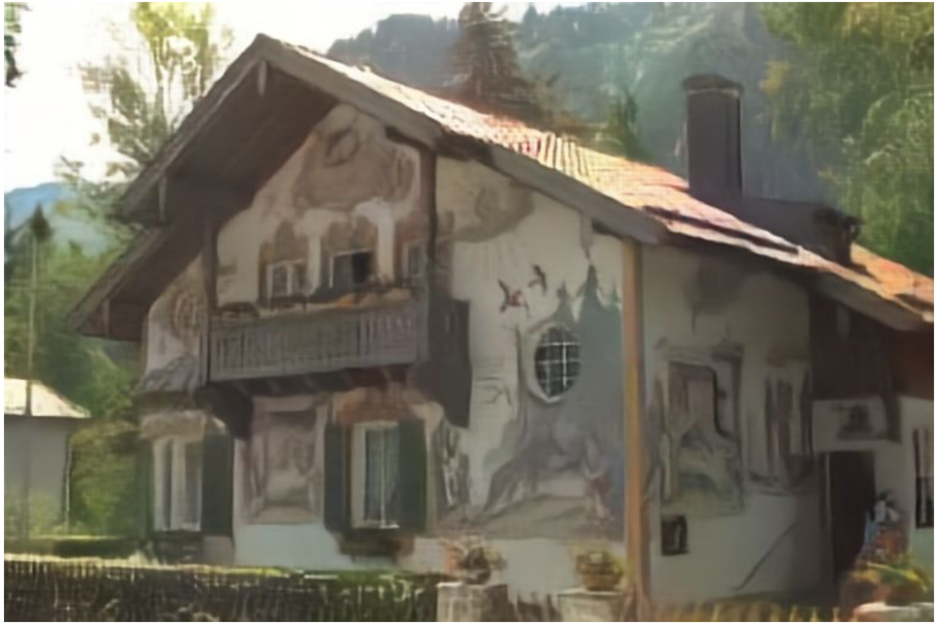
Trained on "Hard" 32x32 Patches



Visual Comparison (JPEG vs Neural Net @ 0.125 bpp)

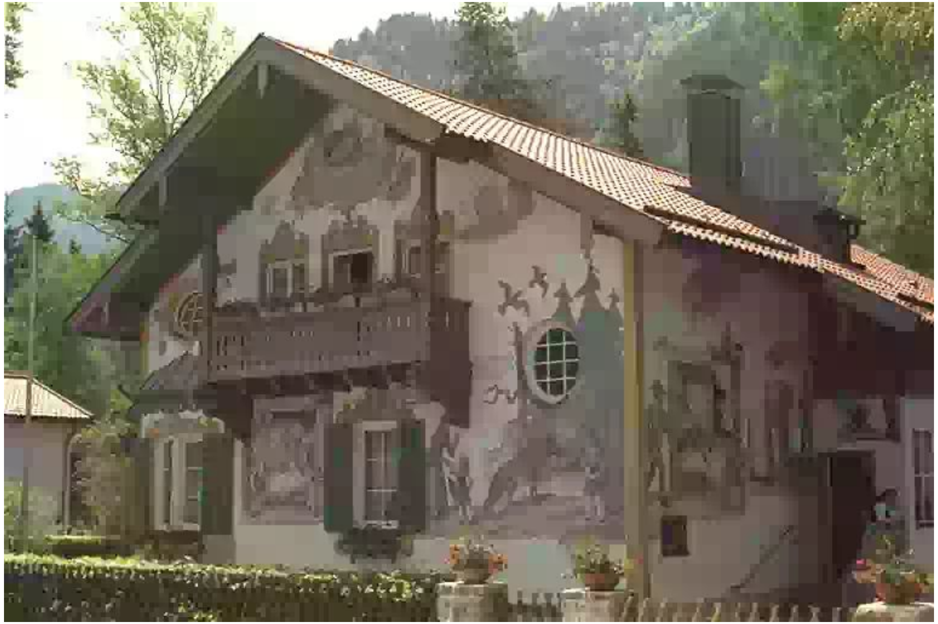


JPEG

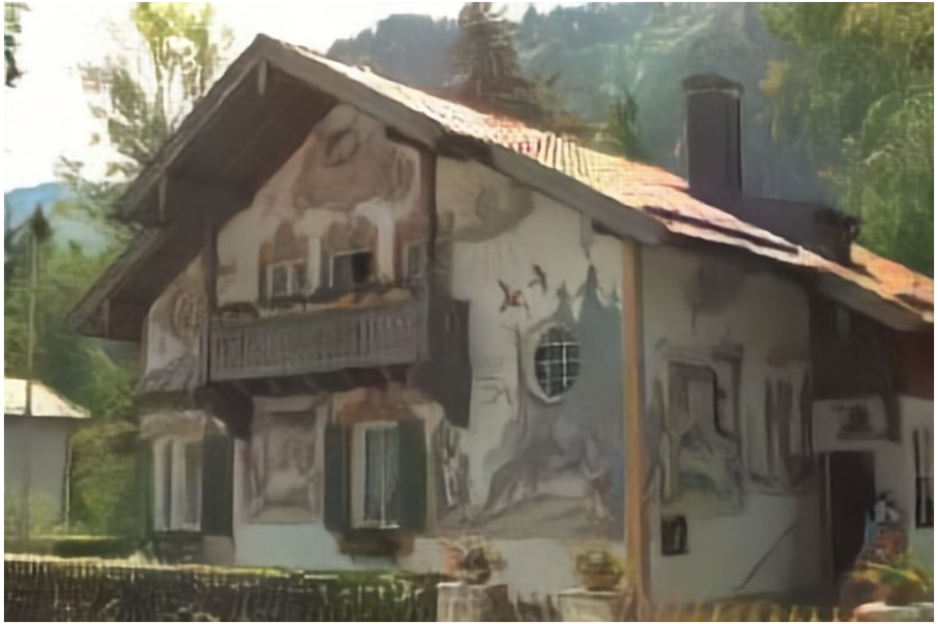


Neural Net (Prime3)

Visual Comparison (WebP vs. Neural Net @ 0.125 bpp)

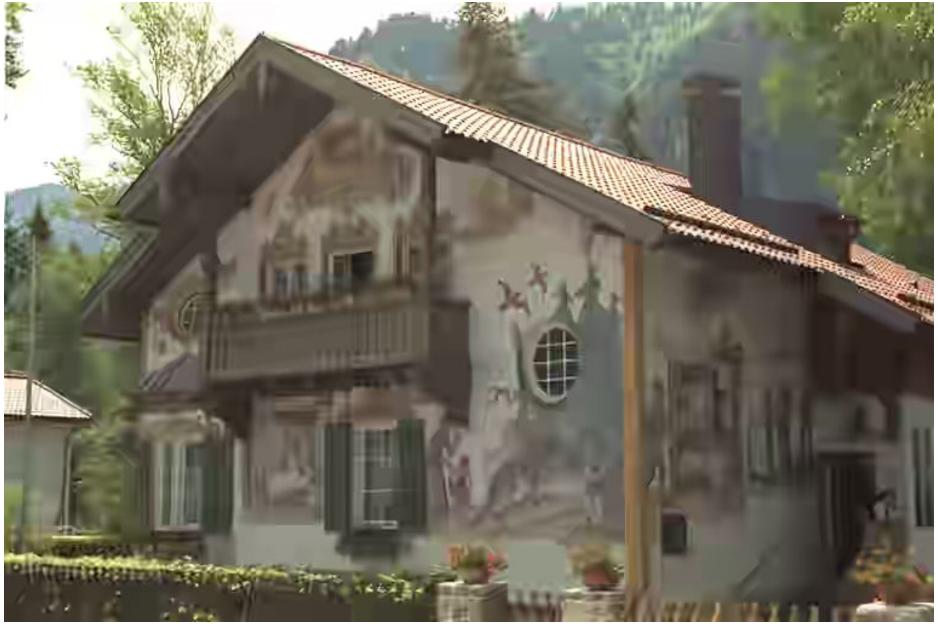


WebP

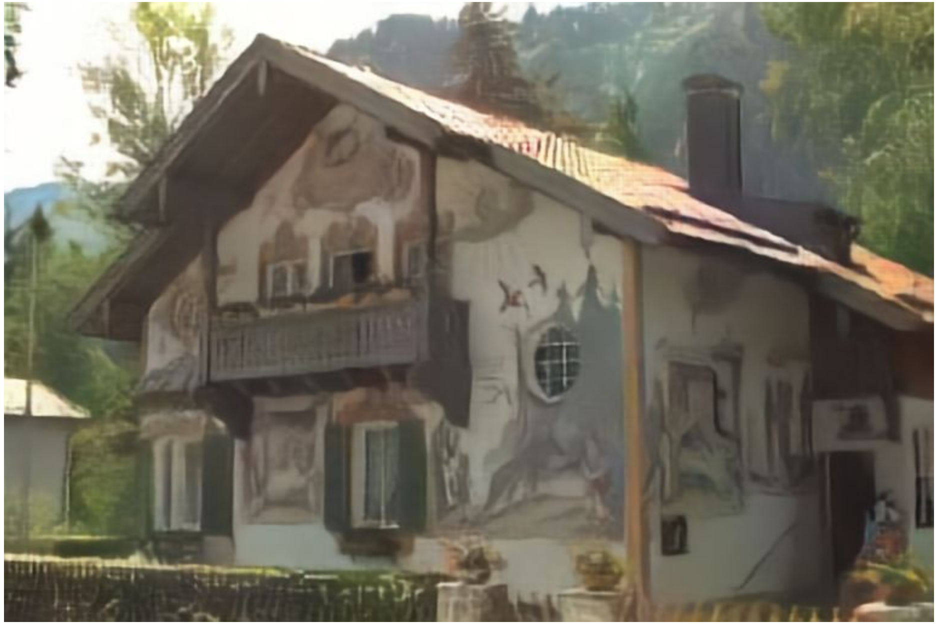


Neural Net (Prime3)

Visual Comparison (BPG vs. Neural Net @ 0.125 bpp)



BPG



Neural Net (Prime3)

Google

Summary of Neural Net Based Compression

- State-of-the-art results: NN architecture outperforms JPEG and WebP
- Devil is in the details:
  - Hard negative mining
  - Entropy coding
  - Novel residual scaling method
- Future work will focus on improving performance (i.e., computation, and quality at a particular bit rate) and exploiting temporal structure (for video).

Questions/Discussion

Questions 1

- To what extent do neural networks improve image compression by high-level knowledge of image semantics?
- Do you believe that CNNs still represent the state of the art for image and vision tasks, or do you believe that more general recurrent architectures will eventually take their place?
- Do you believe that these models for image compression show promise for video compression as well, or are there performance issues when they are applied to this domain?
- What are some other fields besides images you think compression techniques/neural networks can be applied to?
- It seems like many of the papers focus on less standard interests of ML (bitrates, memory usage, thumbnails vs. accuracy, data efficiency). Are there any other metrics in the deep learning which should receive more attention than they do now?
- What decisions go into the tradeoff between test time speed and accuracy versus training time? Is it useful in applications to think about training time as something to optimize?

Questions 2 {Object Detection}

- Do you believe the benchmarks presented in the paper accurately reflect the tradeoffs of a production ML system, and if not then what other factors should be considered when using models in the "real world"?
- Why did you jointly train all models end-to-end using asynchronous gradient updates instead of synchronous?
- In the paper, several techniques are used for encoding bounding boxes. When encoding continuous variables for regression tasks, what are the best practices and why?
- How did you decide to use Faster R-CNN, R-FCN and SSD as your baseline architectures?

Questions 3 {Compression}

- Have you experimented with Binarization techniques other than the one proposed by Williams (1992)? How did these compare and what do you believe are the advantages of the Williams approach?
- It seems like the thumbnails produced by these algorithms are both better looking and take less bits to transmit. Is there a tradeoff in terms of decoding speed or memory usage compared to traditional compression techniques?
- Have you tried the same experiments with non-recurrent cells? Why would LSTMs perform better in this kind of problem?
- Is 'interpretability' of featurization a useful (possibly auxiliary) objective for neural net-based compression?
- Given these 32x32 thumbnails are to be displayed as quickly as possible, decoding speed is surely something to optimize for. How do the decoding speed of the Fully Connected, LSTM, Conv/Deconv, Conv/Deconv LSTM and reference methods (e.g. JPEG) compare on 32x32 images?

Questions 4 {Compression}

- Why do you believe the one-shot models consistently outperformed the scaling and additive models, on average?
- JPEG is a common standard because it isn't computationally expensive (both in memory/time). Is it worth the additional complexity added by using an RNN to do image compression?
- Is there intuition behind why the associative LSTMs were only effective when used in the decoder?
- Did you do any experiments where you added an attention component to your RNN cells? What were the results?
- Is there a similar link to to 'Speed/accuracy trade-offs for modern convolutional object detectors' in terms of satisfying external constraints (resources -> compression rate)? Is this useful?

More information

- Research at Google: http://research.google.com
- Contact: Rahul Sukthankar <sukthankar@google.com>