Deep Learning Literature Overview

Group 5

- Reading Material 2
  - Squeeze and Excitation Networks
  - Faster R-CNN
  - YOLO
  - SSD
- Reading Material 3
  - Show and Tell
  - NMT - Align and Translate
- Reading Material 4
  - Attention is All you Need
  - Image Transformer
- Reading Material 5
  - CycleGAN
  - InfoGAN
  - Self-Attention GANs

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY **DELHI**

Reading Material - 2

Group 5

- Squeeze and Excitation Networks
  - Aman Roy
- Faster R-CNN
  - Gyanesh Anand
- You Only Look Once:Unified, Real-Time Object Detection
  - Pulkit Madaan
- SSD: Single Shot MultiBox Detector
  - Aman Roy

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

SQUEEZE AND EXCITATION NETWORKS

- Interpretation and Analysis
  - Role Of Excitation
    - In early stages of network importance of feature channels is likely to be shared among different classes.
    - At greater depth value of channels becomes much more class specific.
    - At last stage many of the activations are close to 1 and ct as standard residual network. So they can be removed to lower down the number of parameters.
  - Reduction Ratio
    - Performance does not increase monotonically with increased capacity
    - Possibly due to overfitting of channel interdependencies of training set

.

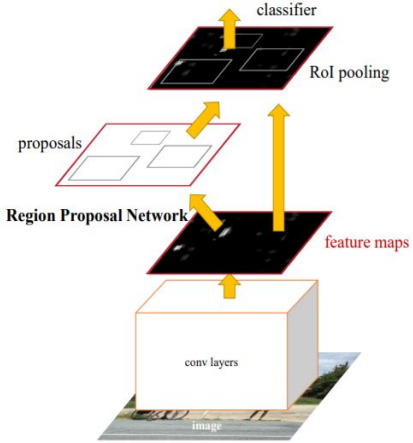
SQUEEZE AND EXCITATION NETWORKS

- Goal
  - Explicitly model the interdependencies between channels of Conv features.
- Need
  - Feature calibration, selectively emphasise important features and suppress less useful one.
  - Attention and Gating mechanism.
- Model (Sequence and Excitation Blocks)
  - Transformation (Convolutional Operator)
  - Squeeze :- Global Average Pooling
  - Excitation :- Adaptive Re-Calibration (flexible,non-mutually exclusive relationship)
    - Parametrise gating mechanism using two FCs.
    - Rescaling transformation outputs using activations.
- SE-Inception and SE-Resnet
  - For SE-Inception, transformation function is entire Inception Module.
  - Flexibility of SE module can easily use Resnet architecture.
  - Small amount of computational overhead.

Faster R - CNN

RCNN ⟹ Fast R-CNN ⟹ Faster R-CNN

- **Need**
  - Fast R-CNN had reduced the running time of Object detection networks
  - But Region proposal computation was as the bottleneck
- **New**
  - **Region Proposal Network (RPN)** to predict proposals
- **Model**
  - Model Consists of 2 modules :-
    - a deep fully convolutional network that proposes regions
    - the Fast R-CNN detector that uses the proposed regions
  - The entire system is a single, unified network for object detection.
  - Using ‘attention’ mechanisms, the RPN module tells the Fast R-CNN module where to look.
  - Jointly train the Fast R CNN and RPN with 4 losses:
    - RPN classify object / not object
    - Regress box coordinates
    - Final classification score
    - Final box coordinates



YOLO: You Only Look Once

- Need
  - Previously Proposal and Detection or repurposed classifiers
  - Complex and slow as separate training
  - Have less global context
- New
  - Single regression problem
  - Image to bounding box (end-to-end)
  - YOLO - 45 fps
  - FAST YOLO - 155 fps
- Model
  - Image to SxS grid
    - Each cell predicts B bboxes(x, y, w, h, objectness)
    - C class probas
  - Final Vector (S, S, B*5+C)
  - Each cell can have only one bbox - Non-maximal suppression
    - Removes redundancy
    - Improves diversity
- Architecture and Loss

SSD (SINGLE SHOT DETECTION)

- Goal
  - A method for detecting objects using a single deep neural network.
- Need
  - Significantly increased speed comes only at the cost of significantly decreased accuracy.
- Model (Sequence and Excitation Blocks)
  - A feed-forward CNN.
  - CNN produces
    - A fixed size collection of bounding boxes and scores for presence of object class in instance boxes.
  - Base Networks.
  - Auxiliary Networks
    - Multi-Scale Feature Maps For detection.
    - Convolutional predictors for detection.
    - Default boxes and Aspect Ratios.

SSD (SINGLE SHOT DETECTION)

- Architecture
  - Matching Strategy
    - which default boxes corresponds to ground truth decision.
  - Training Objective
    - Derived from multibox objective
    - $L(x,c,l,g) = 1/N(Lconf(x,c)+\alpha Lloc(x,l,g))$
  - Choosing scales and aspect ratios for default boxes.
    - The scale of the default boxes for each feature map is computed as:
    - $s_k = s_{min} + (s_{max}-s_{min})(k-1)/(m-1),\ k \in [1,m]$
  - Hard Negative Mining.
  - Data Augmentation.

.

Reading Material - 3

Group 5

- Show and Tell: A Neural Image Caption Generator
  - Aman Roy
- Neural Machine Translation by jointly learning to Align and Translate
  - Pulkit Madaan

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

Show and Tell: A Neural Image Caption Generator

- **Need**
  - Generate natural sentences describing an image.
- **New**
  - Trained to maximize the likelihood of Sentence given Image.
  - Replace encoder RNN by a deep CNN.
- **Model**
  - Maximize the probability (1) of correct description given the image.
  - Applying chain rule to model joint probability (2).
  - Using RNN (specifically LSTM) to model the probability distribution(2).
  - CNN have been used for extracting features from Images.
  - A special start and stop symbols are there which designates the end and start of sentence.
  - Image is input only once at t = -1.
  - Feeding images at every time step can lead to overfitting.
  - Loss is negative log likelihood of correct word at each time step.

$$\theta^\star = \arg\max_\theta \sum_{(I,S)} \log p(S|I;\theta) \qquad (1)$$

$$\log p(S|I) = \sum_{t=0}^{N} \log p(S_t|I, S_0, \ldots, S_{t-1}) \qquad (2)$$

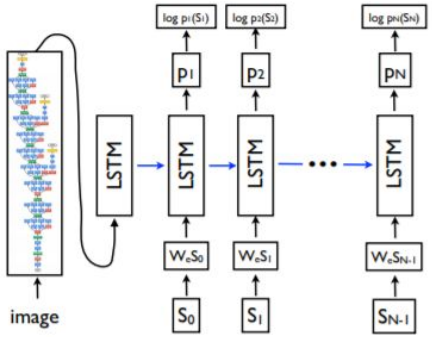
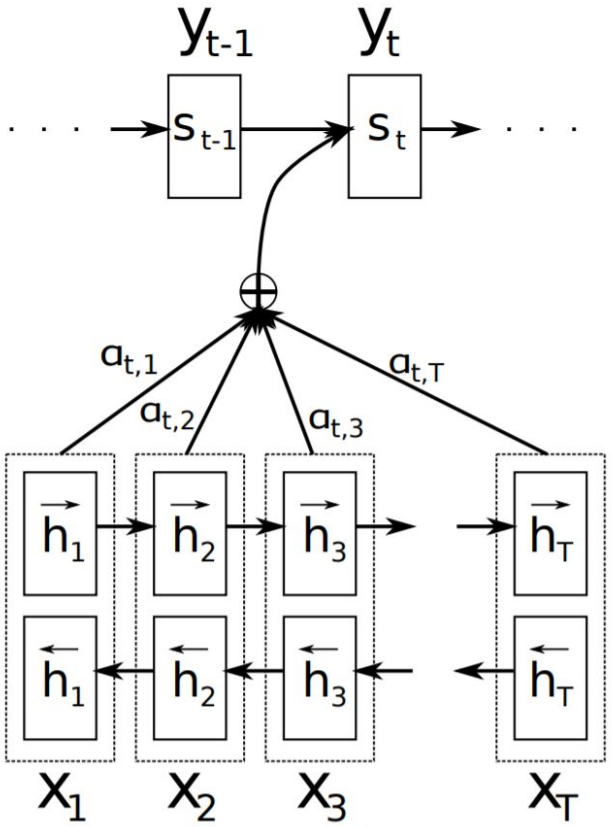


Figure 3. LSTM model combined with a CNN image embedder (as defined in [12]) and word embeddings. The unrolled connections between the LSTM memories are in blue and they correspond to the recurrent connections in Figure 2. All LSTMs share the same parameters.

NMT: Align and Translate

- **Need**
  - Existing models use encoder-decoder RNN
    - fixed length encoding of the input sequence
  - Hence, not able to capture long term dependencies
    - Not even LSTMs
- **New**
  - Use alignment and learn it (attention mechanism)
- **Model**
  - Use BiRNN and concatenate fwd and bwd hidden states(h) at every time step
  - Use previous state information to generate an alignment vector(a)
  - Take a weighted sum of concatenated hidden states w/ alignment vector as weight vector.
  - Use this weighted sum(c), previous state and previous prediction to predict new state(s).
  - Sample new prediction(y) with state as probability vector.



NMT: Align and Translate

- **Model**

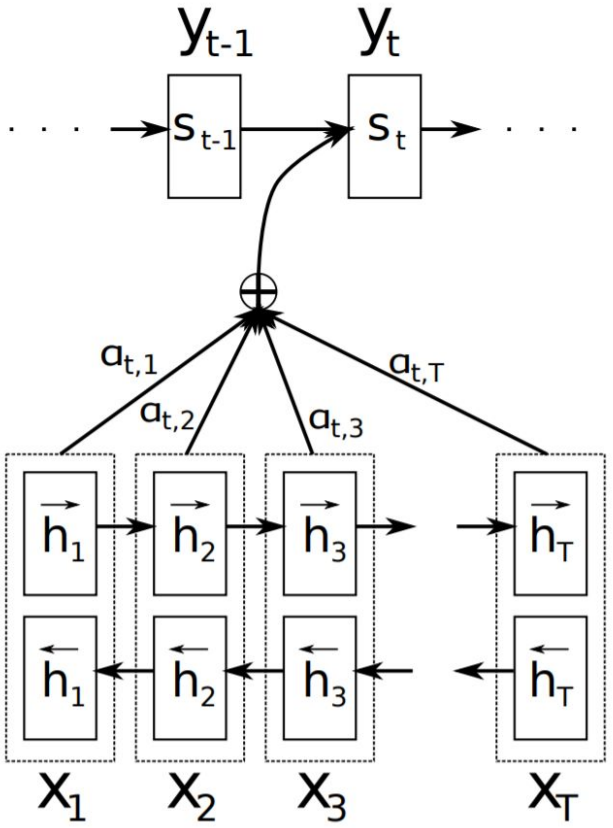
$$p(y_i|y_1,\ldots,y_{i-1},\mathbf{x}) = g(y_{i-1},s_i,c_i)$$

$$s_i = f(s_{i-1},y_{i-1},c_i)$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1},h_j)$$



Reading Material - 4

Group 5

- Attention is All you Need
  - Pulkit Madaan
- Image Transformer
  - Aman Roy

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

Attention is All you Need

- **Need**
  - Existing models are complex CNNs or RNNs
  - Factoring sequential nature leads to loss of parallelism.
- **New**
  - Use only attention:
    - Self-Attention
    - Encoder-Decoder Attention
  - Multi-headed attention
  - Use positional encodings
  - Multiple encoders and decoders
- **Model**
  - Encoder
    - Add positional to input embeddings
    - Use multi-headed self attention with normalised residual connection
    - Feed forward the outputs parallely
    - Feed Forward has normalised residual connections
    - Feed Forward produces Key and Value Matrices for Enc-Dec Attention

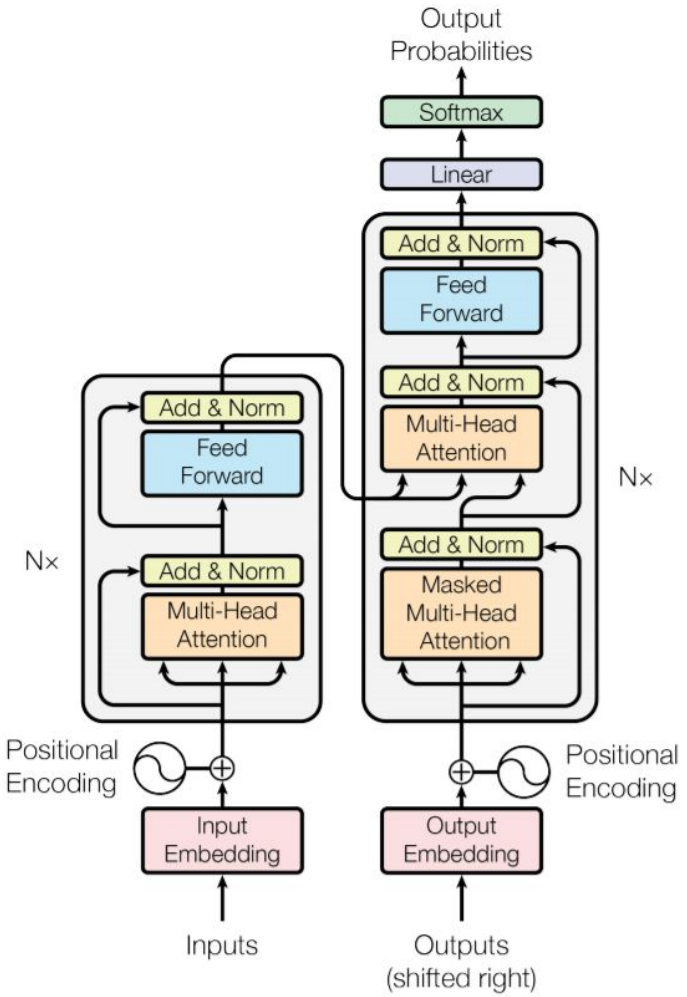


Figure 1: The Transformer - model architecture.

Attention is All you Need

- **Model**
  - Decoder
    - Add positional to masked translated embeddings
    - Use multi-headed self attention with normalised residual connection
    - Generate the Context Matrix for Enc-Dec Attention
    - Get Query and Value Matrix from Encoder for Enc-Dec Attention
    - Apply multi-headed enc-dec attention with normalised residual connection
    - Feed forward the outputs parallely
    - Feed Forward has normalised residual connections
    - Feed Forward → Linear → Softmax
    - Sample from softmax probabilities for next translated word
    - Add this word to masked translated embeddings.

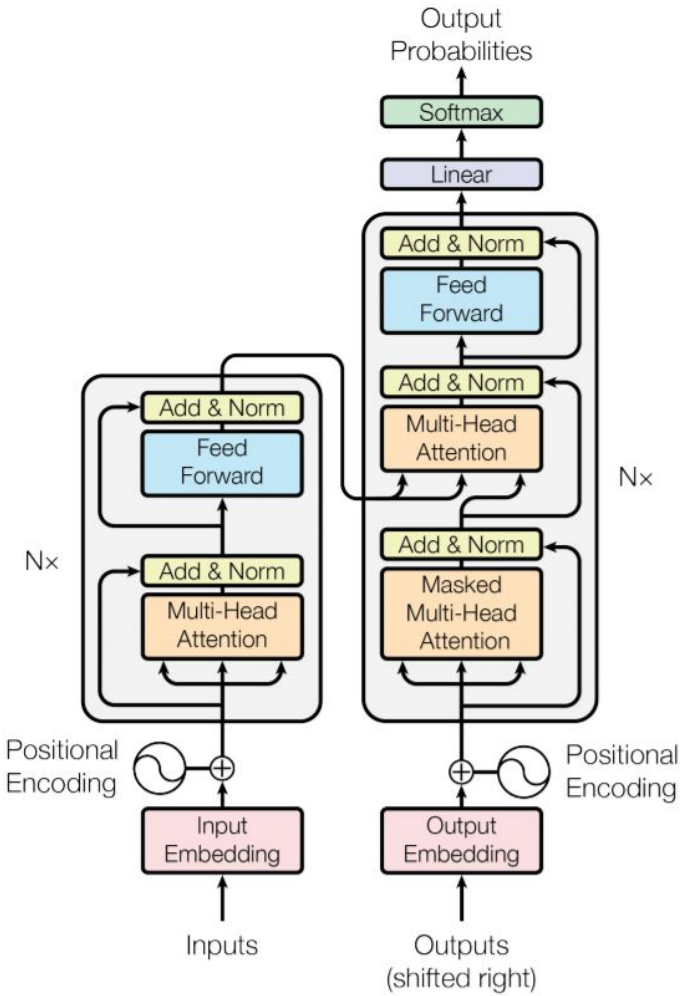
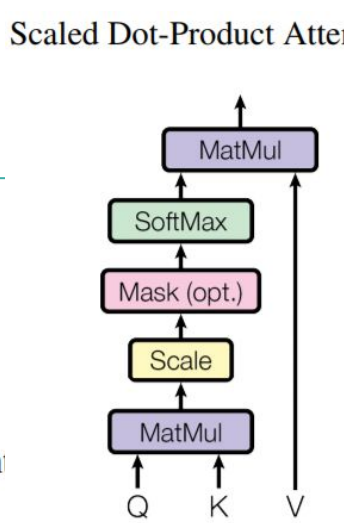


Figure 1: The Transformer - model architecture.

Attention is All you Need

- **Model**
  - Multi-headed Self Attention
    - Multiple Self Attention Modules
  - Self Attention
    - Takes word+positional encodings
    - Calculates Query(Q), Key(K) and Value(V) matrices.
    - Weights Every word w.r.t every other word using Q, K, V
    - Gives out a weighted input context vector

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



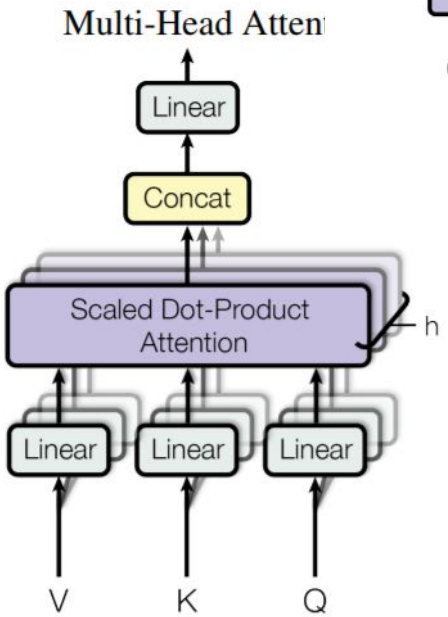


Image Transformer

- **Need**
  - Balance between RF of pixelRNN and pixelCNN.
- **New**
  - Proposed a model entirely based upon Self Attention Network.
  - Proposed locally restricted form of multi-head self attention.
- **Model**
  - IMAGE REPRESENTATION
    - Treat each pixel as discrete categories or ordinal values.
    - Ran 1*3 window size, 1*3 strided convolution to combine 3 channels per pixel.
    - Positional encoding(SIne and Cosine) on each pixel representation.
  - Self-Attention
    - Used encoder-decoder architecture.
    - Both encoder and decoder uses stack of self attention.
    - Decoder :- additional attention mechanism for encoder representation

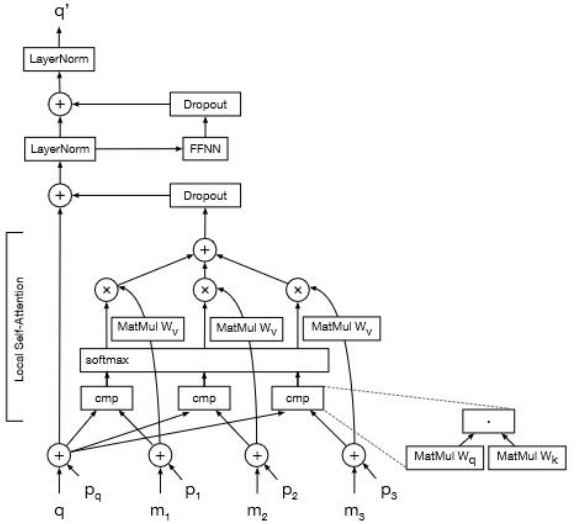


Figure 1. A slice of one layer of the Image Transformer, recomputing the representation q' of a single channel of one pixel q by attending to a memory of previously generated pixels $m_1, m_2, \ldots$. After performing local self-attention we apply a two-layer position-wise feed-forward neural network with the same parameters for all positions in a given layer. Self-attention and the feed-forward networks are followed by dropout and bypassed by a residual connection with subsequent layer normalization. The position encodings $p_q, p_1, \ldots$ are added only in the first layer.

Image Transformer

- **Model**
  - Self-Attention
    - For a position, it computes position's current (Q) representation with other position's representation (M) to get attention weights.
    - FCC along with RELU is applied on resulting vector.
    - Used dropout, merged residual connections and layer normalization.
  - Local Self Attention
    - Difficulty in applying self-attention on decoder's output.
    - Restricting M to capture local neighborhood around query.
    - Partition image into query block -> associate with larger memory block(with query block) -> self attention computed in parallel.
    - Two schemes for query and their memory block
      - Local Attention
        - Positional encodings in raster scan order

$$q_a = \text{layernorm}(q + \text{dropout}(\text{softmax}\left(\frac{W_q q (M W_k)^T}{\sqrt{d}}\right) M W_v)) \quad (1)$$

$$q' = \text{layernorm}(q_a + \text{dropout}(W_1 \text{ReLu}(W_2 q_a))) \quad (2)$$

Image Transformer

- **Model**
  - Local Self Attention
    - Two schemes for query and their memory block
      - 1D Local Attention
        - Partition into non-Overlapping query blocks Q.
        - Build memory block M from Q.
        - Additional l_m position corresponding to pixels that have been generated.
      - 2D Local Attention
        - Partition input tensor with PE in rectangular query blocks.
        - Generate image ordered by query.
- **LOSS**
  - Used MLE.
  - Experiment :- Categorical distribution across channel and discretized logistics over three channel.

$$\log p(x) = \sum_{t=1}^{h \cdot w \cdot 3} \log p(x_t \mid x_{<t})$$

Reading Material - 5

Group 5

- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
  - Pulkit Madaan
- InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
  - Aman Roy
- Self-Attention Generative Adversarial Networks
  - Gyanesh Anand

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

Cycle GAN

- **Need**
  - All image to image translations require explicit mapping between source and target images
  - This is rarely possible in real life
  - Some models assume X and Y lies in same low dimensional space.
  - Some models assume a similarity
- **New**
  - Learn both the mapping from source to target as well as its inverse via cycle consistency.
- **Model**
  - G tries to generate images from Y with X as input and tries to fool Dy
  - F tries to generate images from X with Y as input and tries to fool Dx
  - Discriminator is trained for both X and Y

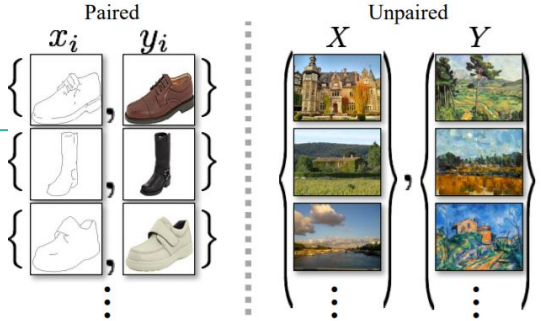
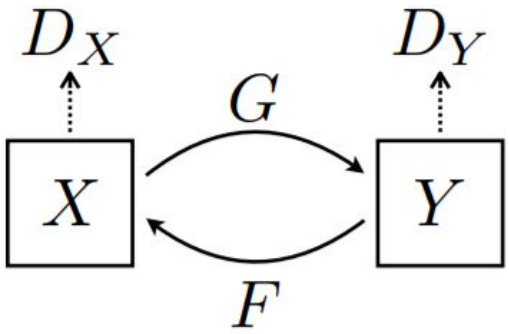


Figure 2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between x_i and y_i exists [22]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^N$ $(x_i \in X)$ and a target set $\{y_j\}_{j=1}$ $(y_j \in Y)$, with no information provided as to which x_i matches which y_j.



$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

Cycle GAN

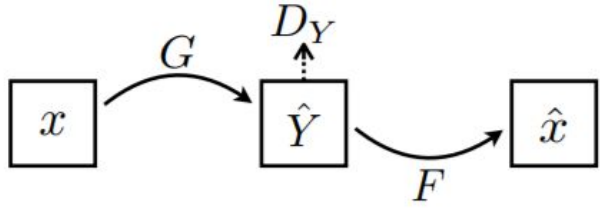
- **Model**
  - In order to strengthen and supervise the training better cycle consistency is used
  - We try to achieve F(G(x)) ≃ x and G(F(y)) ≃ y

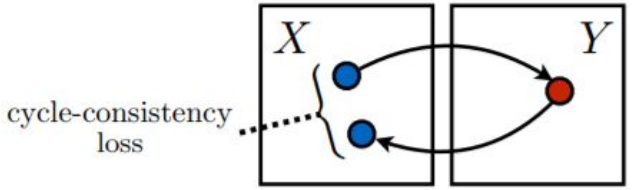
$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

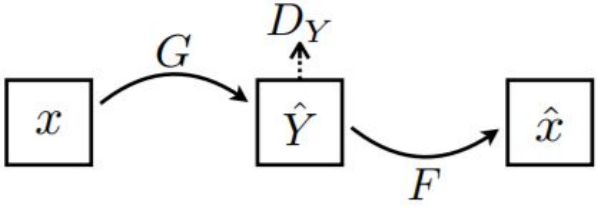
Our full objective is:

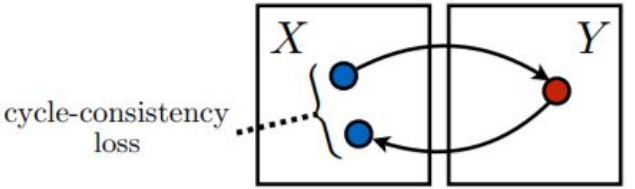
$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

$$G^*, F^* = \arg \min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y)$$









InfoGAN

- **Need**
  - Learning Disentangled Representation that is explicitly represent salient attributes of data instance.
- **New**
  - Maximize mutual information between GAN's noise variables and observations
- **Model**
  - Generative Adversarial Network (GAN).
    - Min-max objective function.
  - Mutual Information for Inducing Latent Codes
    - Continuous input noise vector :- can lead to entangling of the features learned by generator and each component of z does not correspond to semantic feature of data.
    - Decompose input noise vector into two parts
      - z :- source of incompressible noise.
      - c :- latent code targeting salient features.
    - High mutual information between c and G(z,c) (generator distribution) i.e. high I(z,G(z,c)).

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

    - Loss Function :- $\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c;G(z,c))$

InfoGAN

- **Model**
  - Variational Mutual Information Maximization
    - I(c,G(z,c)) is difficult to maximize as it requires P(c|x).
    - Obtaining lower bound using Variational Information Maximization(4).

$$
\begin{aligned}
I(c; G(z,c)) &= H(c) - H(c|G(z,c)) \\
&= \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log P(c'|x)]] + H(c) \\
&= \mathbb{E}_{x\sim G(z,c)}[\underbrace{D_{\mathrm{KL}}(P(\cdot|x) \parallel Q(\cdot|x))}_{\geq 0} + \mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \qquad (4) \\
&\geq \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c)
\end{aligned}
$$

- **Model**
  - FINAL LOSS FUNCTION

$$
\min_{G,Q} \max_{D} V_{\mathrm{InfoGAN}}(D,G,Q) = V(D,G) - \lambda L_I(G,Q)
$$

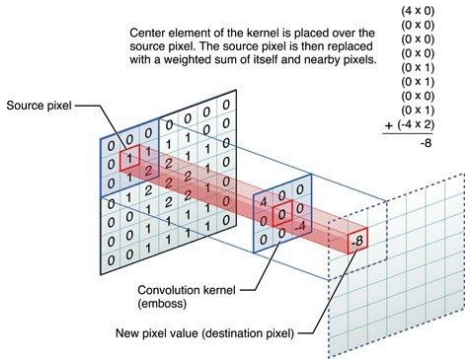
Self-Attention GANs

Problems with Existing ImageNet GANs

- Good at generating images with few structural constraints (ocean, sky & landscape) i.e. easier to generate images with simpler geometry
- Failed on images with some specific geometry like dogs, horses and many more
- Fails at capturing geometric or structural patterns
    - Dogs with good fur, but without two distinct legs

Limitations Of Convolutional GANs

- Problem is arising because the convolution is a local operation so the receptive fields may not be large enough to cover larger structures
- Convolutional Kernel Size could be increased but it becomes slower than and GANs become harder to train



https://arxiv.org/pdf/1805.08318.pdf

Self-Attention GANs

Self-Attention GANs

- The solutions to keeping computational efficiency and having a large receptive field at the same time is Self-Attention.
- Helps create a balance between efficiency and long-range dependencies (= large receptive fields) by utilizing the attention mechanism
- Self Attention is used both in Generator And Discriminator

Model

- We first compute the attention map β and than compute self-attention output
- **ATTENTION MAP :** Multiply **x** with W_f and W_g (these are model parameters to be trained) and use them to compute the attention map β

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{W_g x} \qquad \boldsymbol{x} \in \mathbb{R}^{C \times N}, \boldsymbol{W_f} \in \mathbb{R}^{\bar{C} \times C}, \bar{C} = C/8$$

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W_f x} \qquad \boldsymbol{W_g} \in \mathbb{R}^{\bar{C} \times C}$$

$$s_{ij} = \boldsymbol{f}(\boldsymbol{x_i})^T \boldsymbol{g}(\boldsymbol{x_j})$$

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})} \qquad \beta \in \mathbb{R}^{N \times N}$$

$\beta_{j,i}$ indicates the extent to which the model attends to the i^{th} location when synthesizing the j^{th} region.

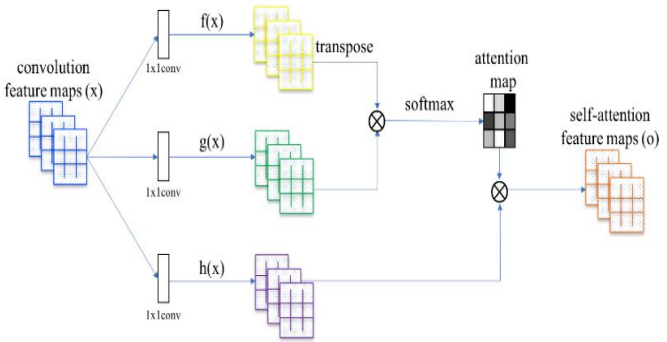


Figure 2: The proposed self-attention mechanism. The ⊗ denotes matrix multiplication. The softmax operation is performed on each row.

Self-Attention GANs

- **SELF-ATTENTION output :** Multiply x with W_h (model parameters to be trained also) and merge it with the attention map β to generate the self-attention feature map output o.

$$h(x_i) = W_h x_i \qquad W_h \in \mathbb{R}^{C \times C}$$
$$o_j = \sum_{i=1}^{N} \beta_{j,i} h(x_i) \qquad o \in \mathbb{R}^{C \times N}$$

- Convolutional layer gives the output : $O = y * o_i + x_i$
  - Initially the y_i is assigned to 0 This allows the network to first rely on the cues in the local neighborhood– since this is easier – and then gradually learn to assign more weight to the non-local evidence.

Loss Function

- The loss function used is just the hinge version of the adversarial loss.

$$L_D = -\mathbb{E}_{(x,y)\sim p_{data}}[\min(0, -1 + D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_{data}}[\min(0, -1 - D(G(z),y))],$$
$$L_G = -\mathbb{E}_{z\sim p_z, y\sim p_{data}} D(G(z),y),$$

Result

- The metrics used in the paper are Inception Score (IS, higher is better) and Frechet-Inception Distance(FID, lower is better).
- Beats previous state of art by goodmargin
- To improve the training, different learning rates are used for the discriminator and the generator (called TTUR in the paper). In addition, spectral normalization (SN) is used to stabilize the GAN training.

Model	Inception Score	FID
AC-GAN [31]	28.5	/
SNGAN-projection [17]	36.8	27.62*
SAGAN	**52.52**	**18.65**

Supplementary Material

SQUEEZE AND EXCITATION NETWORKS (1/3)

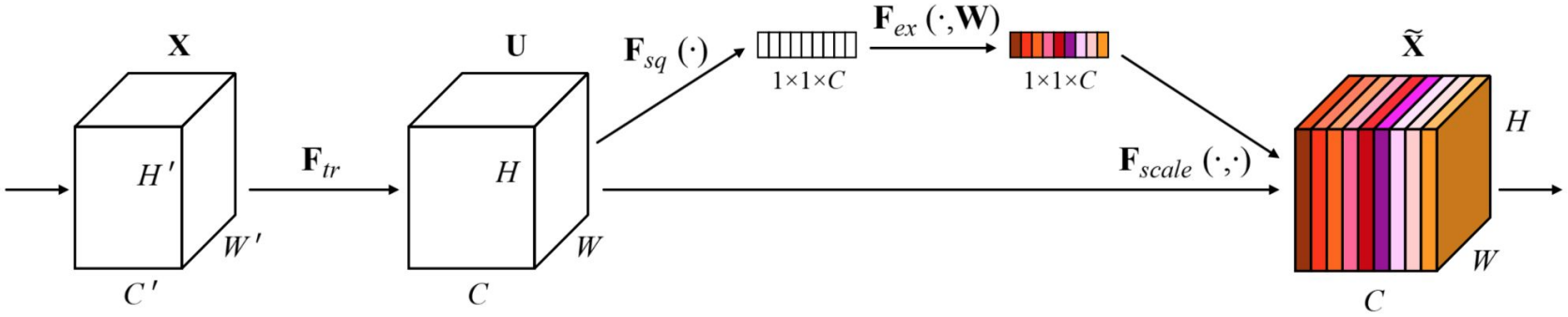
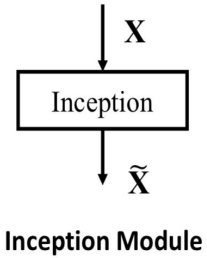
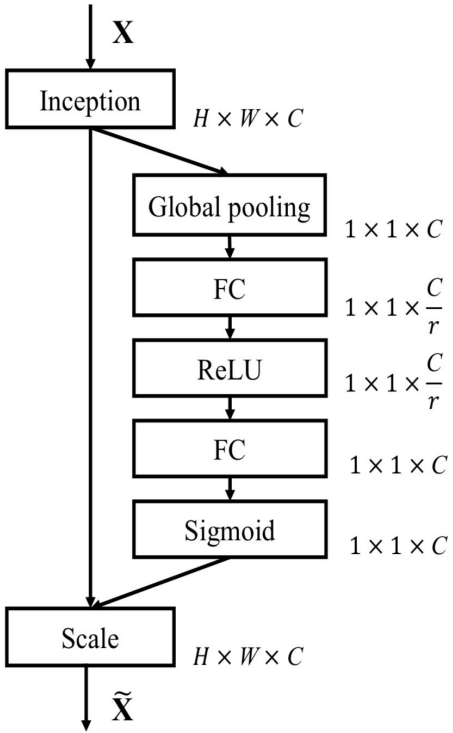


Figure 1: A Squeeze-and-Excitation block.

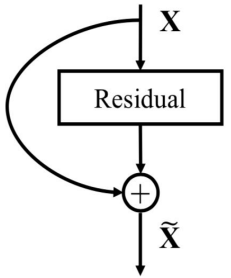
SQUEEZE AND EXCITATION NETWORKS (2/3)



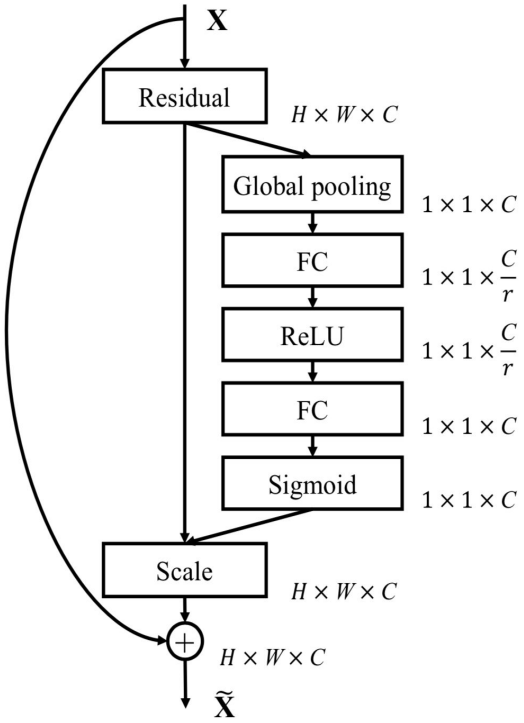
Inception Module



SE-Inception Module



ResNet Module



SE-ResNet Module

SQUEEZE AND EXCITATION NETWORKS (3/3)

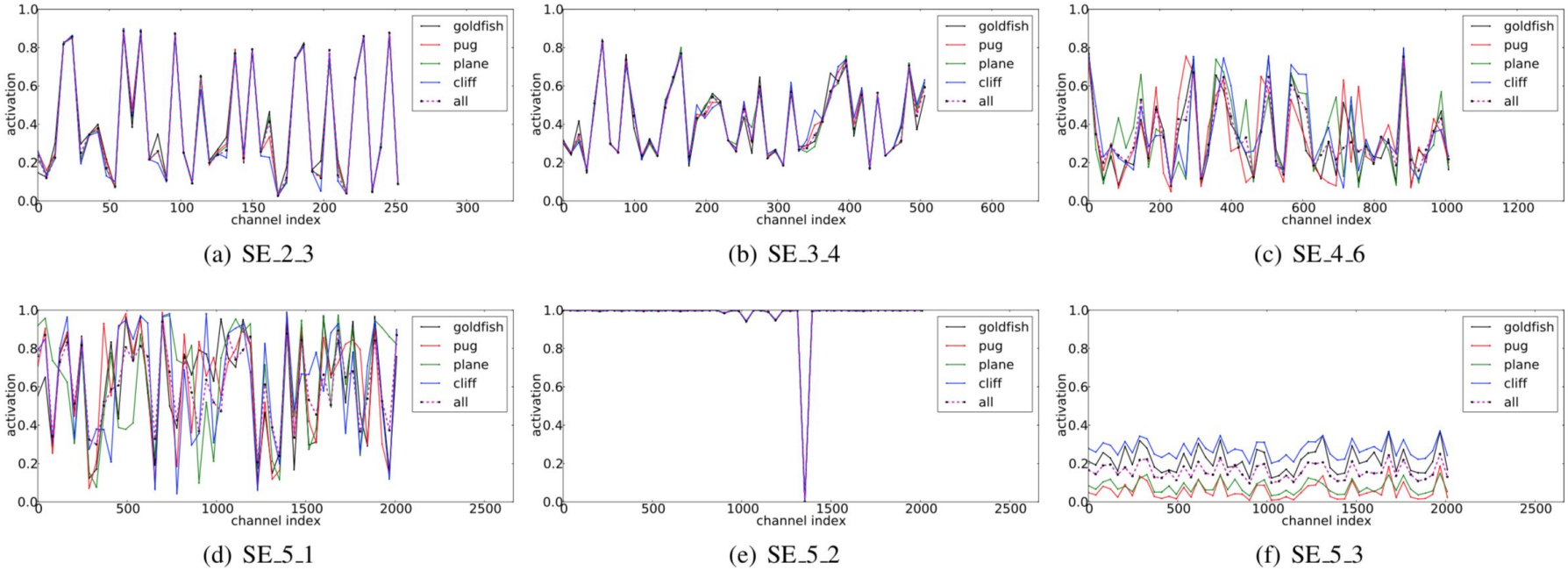


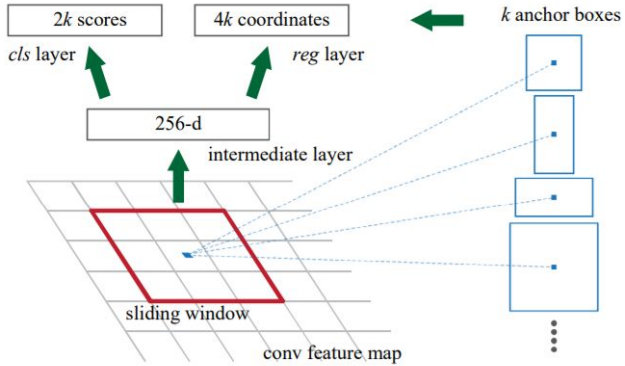
Figure 5: Activations induced by *Excitation* in the different modules of SE-ResNet-50 on ImageNet. The module is named as "*SE_stageID_blockID*".

Faster R-CNN - Supplementary Slides (1/5)

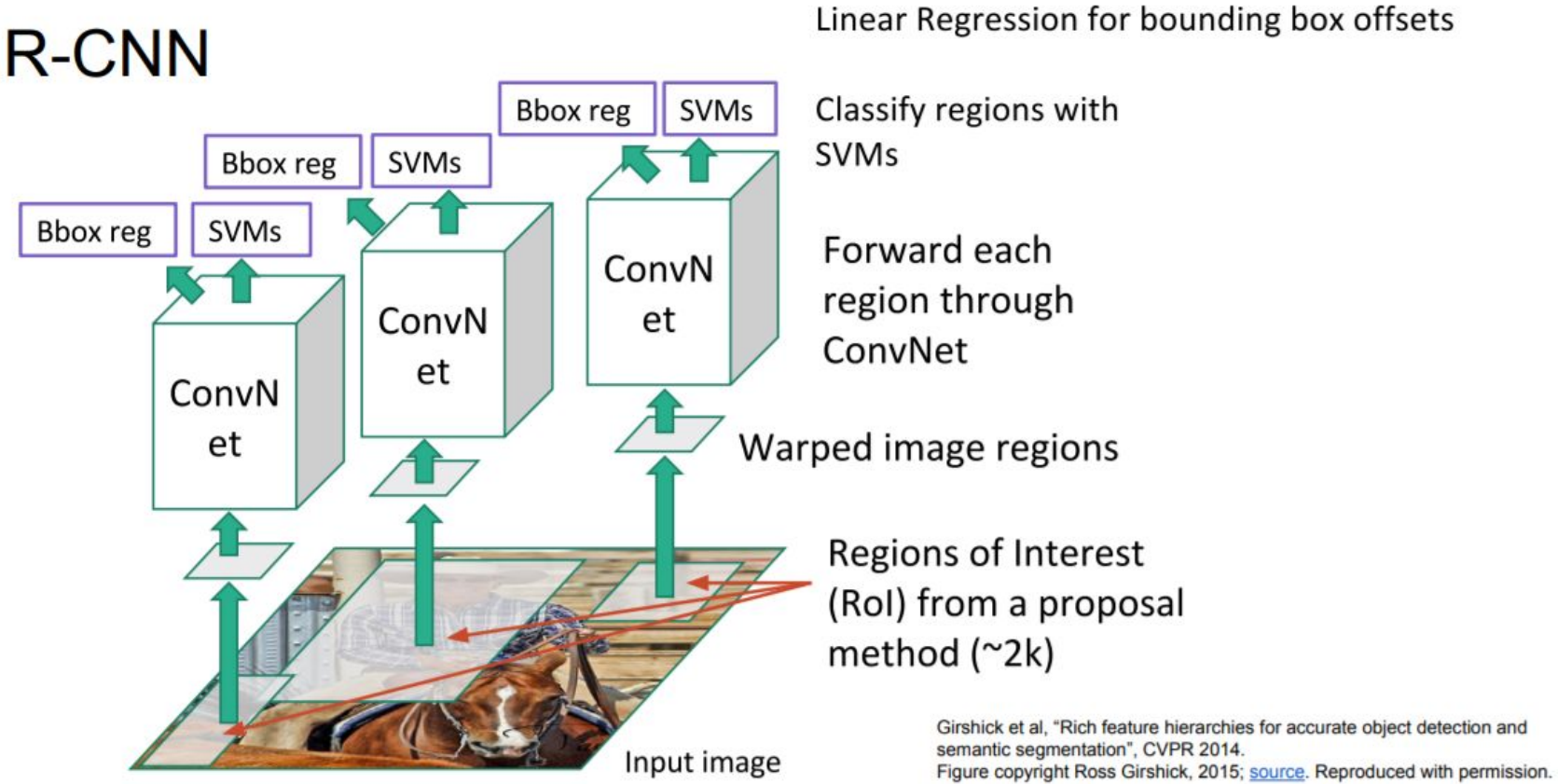
- ➔ Region Of Interest (ROI) classically proposed by Selective Search Algorithm. It takes about 2 seconds for that.
- ➔ But Fast R CNN can detect the object from a proposed region in just 0.5 seconds
- ➔ So Finding Region Of Interest is the bottleneck.
- ➔ The paper on Faster R CNN proposes to reduce the Region Proposal time using RPN.
- ➔ RPN would function alongside the Fast R CNN
- ➔ Combination is done using the Attention Model giving more attention to the ROI.

RPN

- ➢ Construct an RPN by adding a few additional convolutional layers that simultaneously regress region bounds and objectness scores at each location on a regular grid.
- ➢ The RPN is thus a kind of fully convolutional network (FCN) and can be trained end-to end specifically for the task for generating detection proposals
- ➢ Uses Anchors

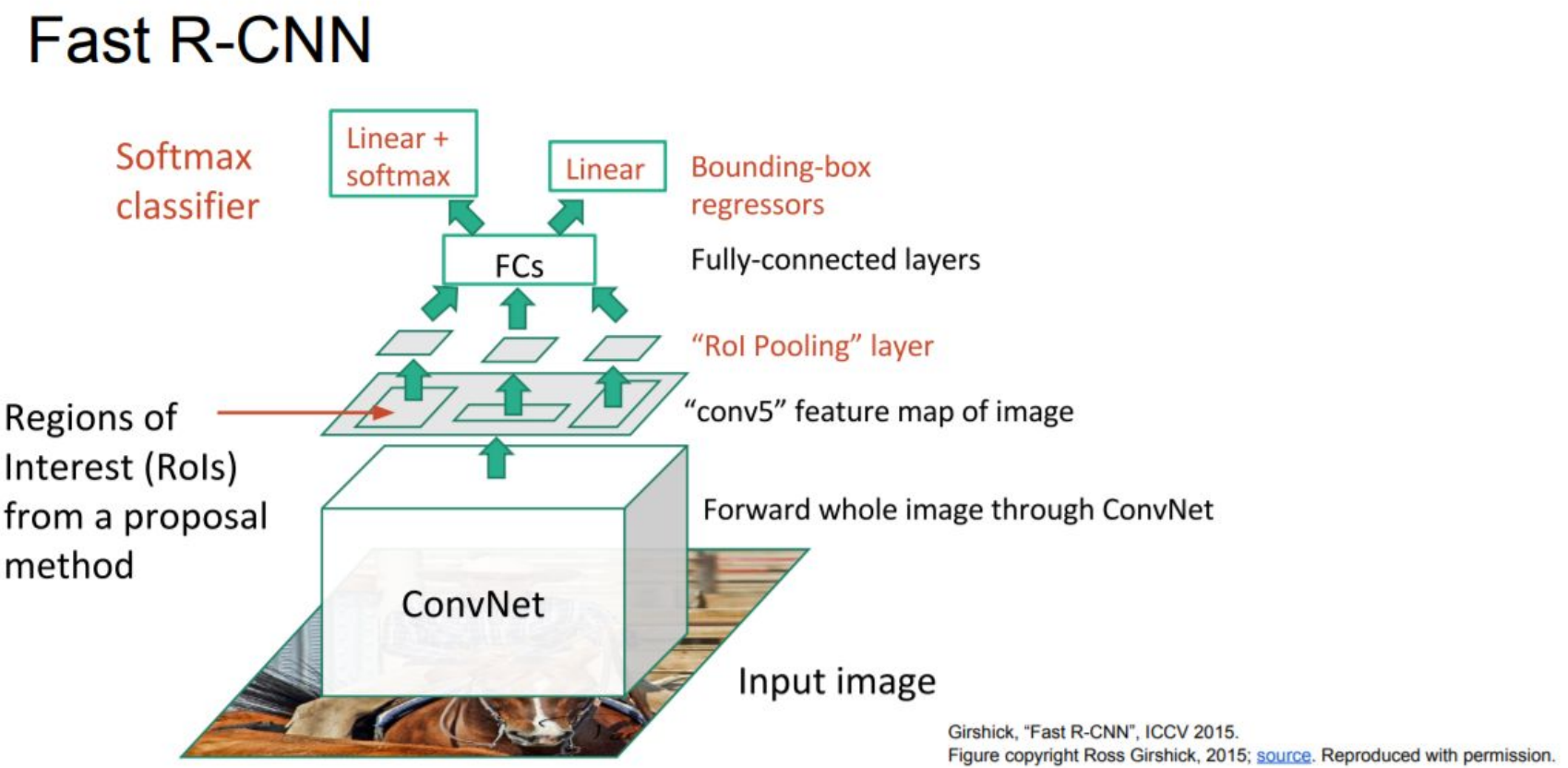


R-CNN (2/5)



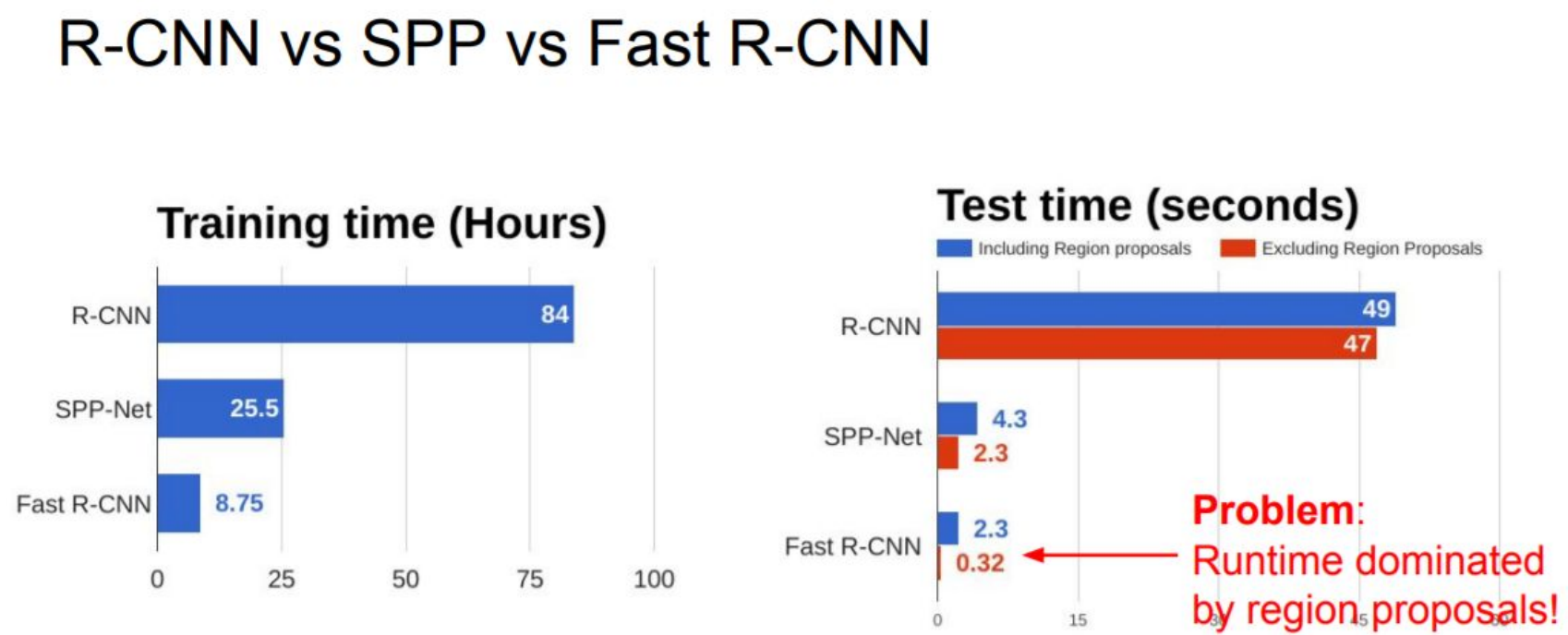
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN (3/5)



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Run-Time Comparison (4/5)

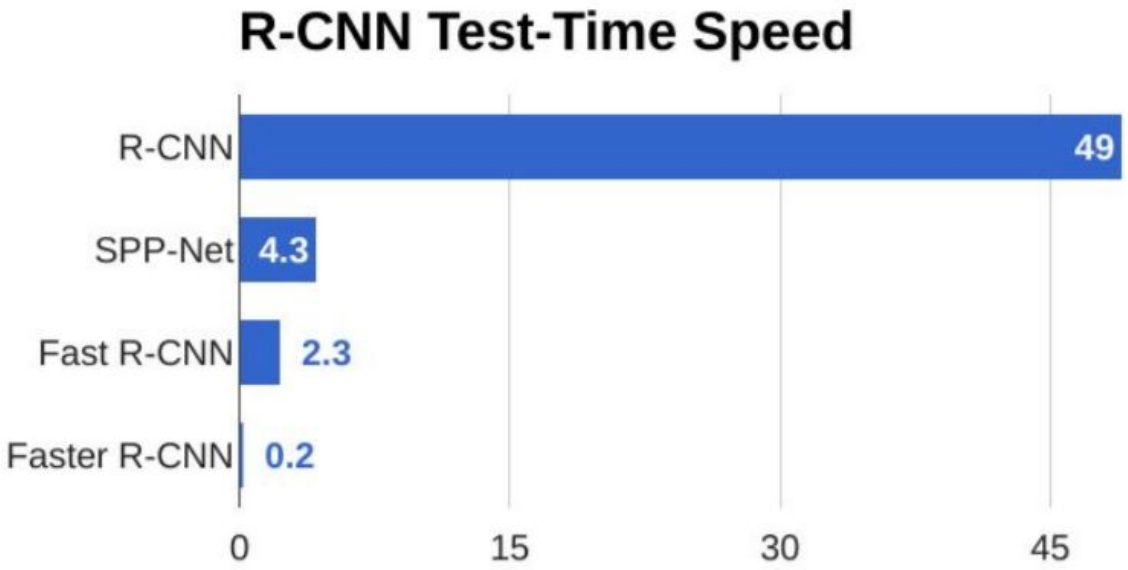


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Run-Time Comparison (5/5)

Faster R-CNN:

Make CNN do proposals!



YOLO : You Only Look Once - LOSS (1/2)

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

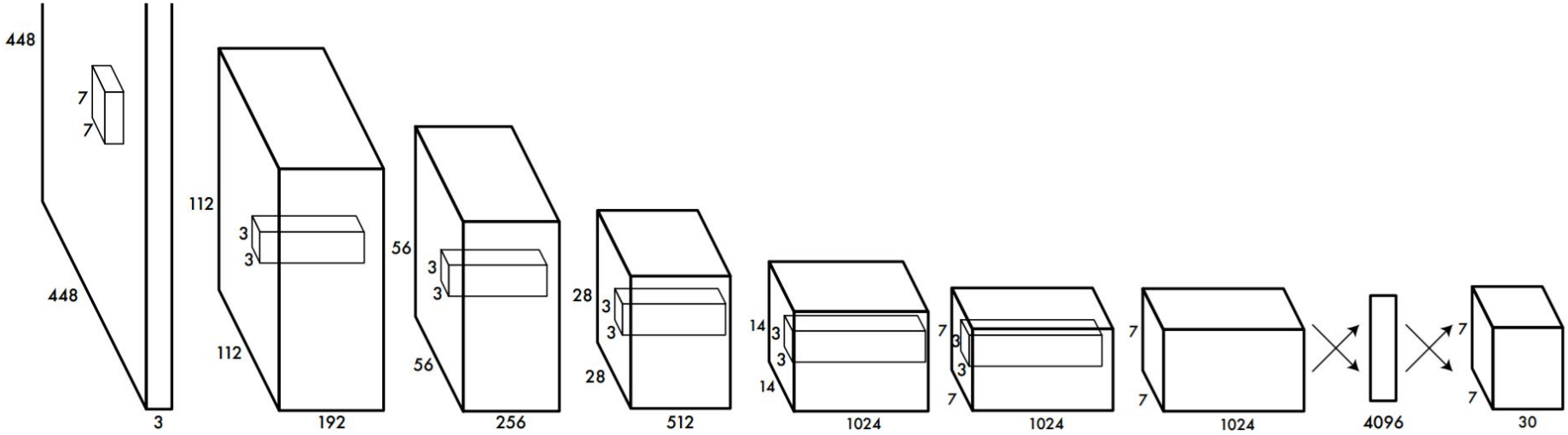
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

YOLO : You Only Look Once - ARCH (2/2)



SSD (1/2)

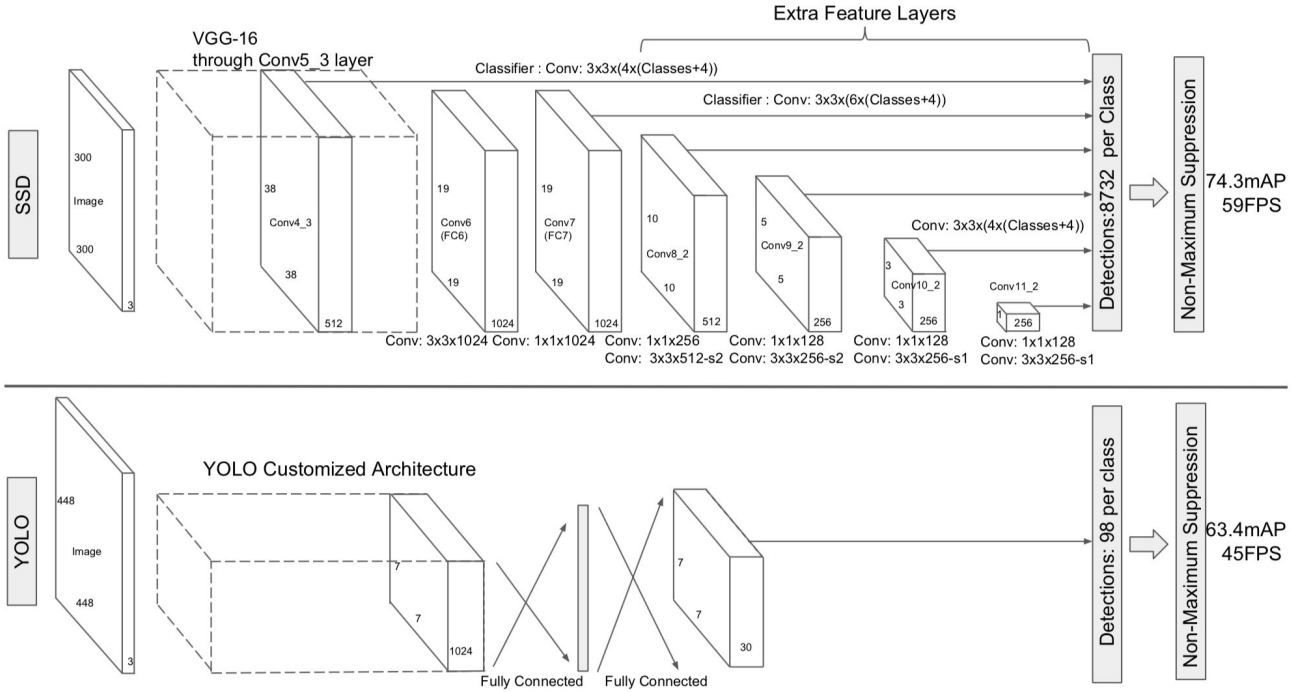
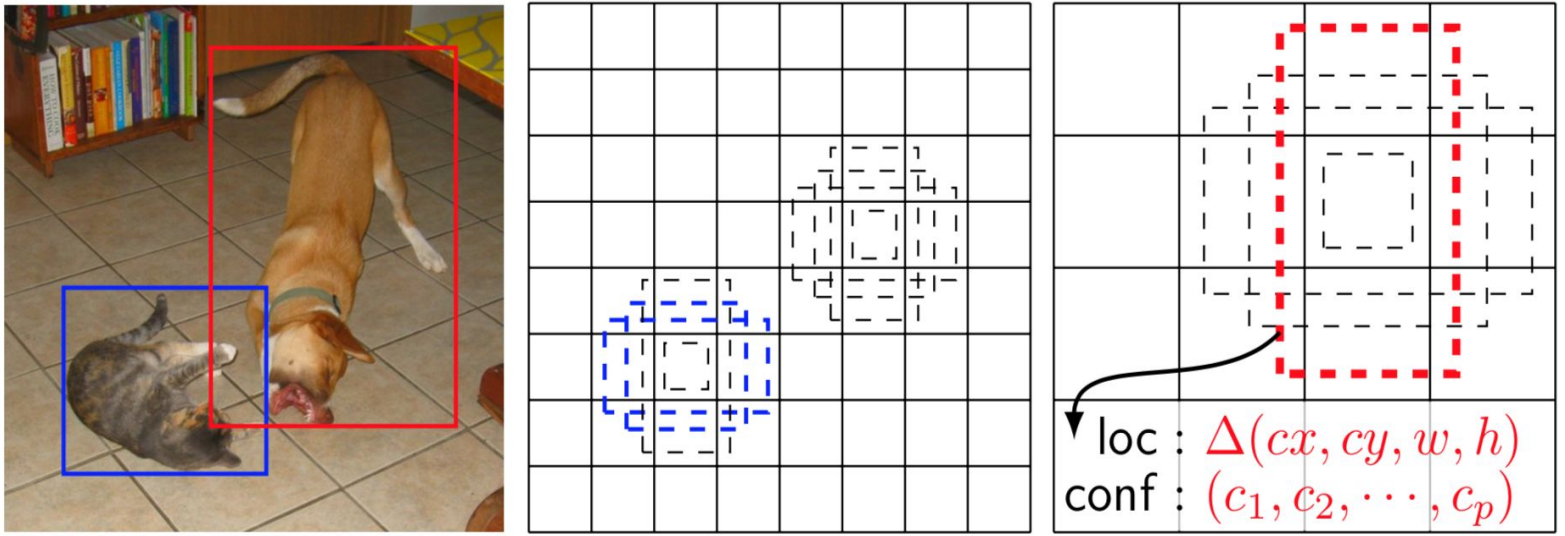


Fig. 2: A comparison between two single shot detection models: SSD and YOLO

SSD (2/2)



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Show and Tell (1/2)

Figure 5. A selection of evaluation results, grouped by human rating.

Show and Tell (2/2)

Metric	BLEU-4	METEOR	CIDER
NIC	**27.7**	**23.7**	**85.5**
Random	4.6	9.0	5.1
Nearest Neighbor	9.9	15.7	36.5
Human	21.7	25.2	85.4

Table 1. Scores on the MSCOCO development set.

Approach	PASCAL (xfer)	Flickr 30k	Flickr 8k	SBU
Im2Text [24]				11
TreeTalk [18]				19
BabyTalk [16]	25			
Tri5Sem [11]			48	
m-RNN [21]		55	58	
MNLM [14][5]		56	51	
SOTA	25	56	58	19
NIC	**59**	**66**	**63**	**28**
Human	69	68	70	

Table 2. BLEU-1 scores. We only report previous work results when available. SOTA stands for the current state-of-the-art.

NMT: Align and Translate (1/3)

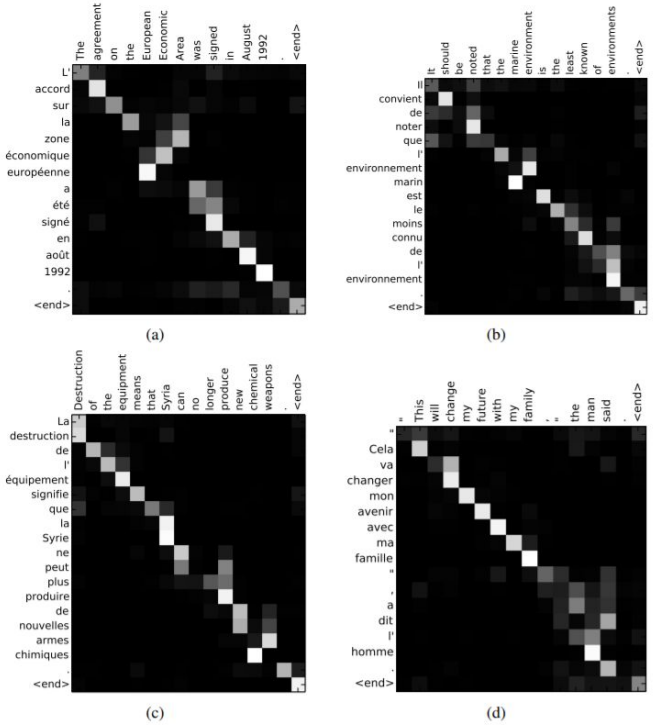


Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an arbitrary sentence. (b–d) three randomly selected samples among the sentences without any unknown words and of length between 10 and 20 words from the test set.

NMT: Align and Translate (2/3)

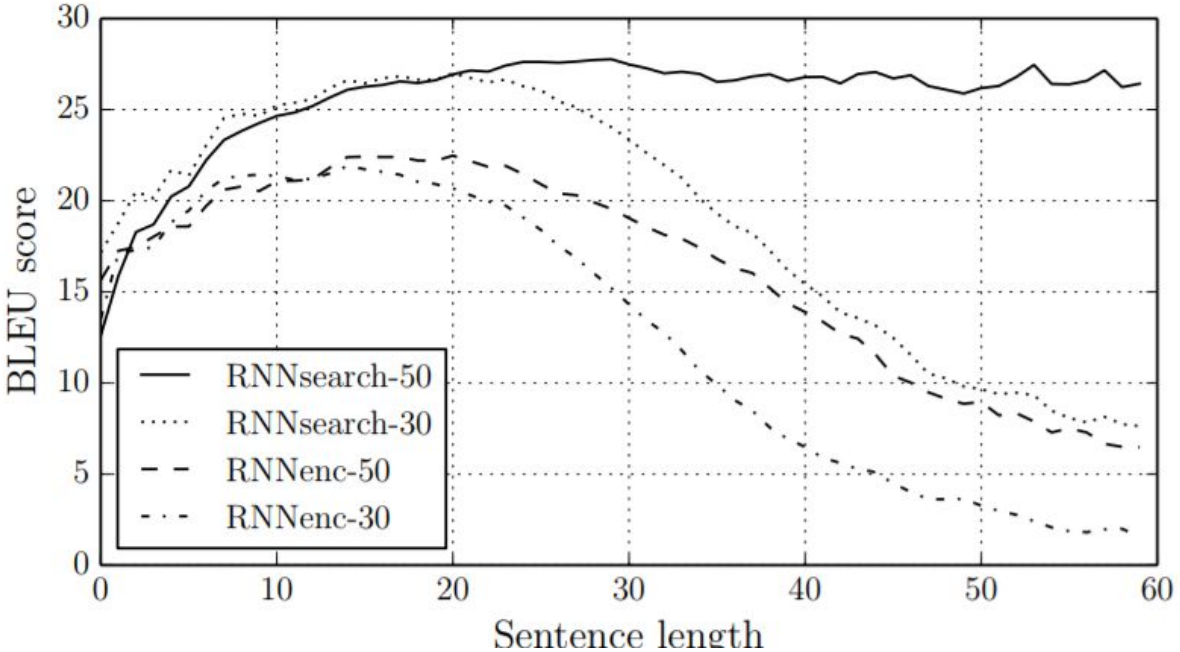


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

NMT: Align and Translate (3/3)

Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50⋆	28.45	36.15
Moses	33.30	35.63

Table 1: BLEU scores of the trained models computed on the test set. The second and third columns show respectively the scores on all the sentences and, on the sentences without any unknown word in themselves and in the reference translations. Note that RNNsearch-50⋆ was trained much longer until the performance on the development set stopped improving. (○) We disallowed the models to generate [UNK] tokens when only the sentences having no unknown words were evaluated (last column).

Attention is All you Need (1/2)

Positional Encodings

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [30]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

Attention is All you Need (2/2)

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

In Table 3 rows (B), we observe that reducing the attention key size d_k hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings [8], and observe nearly identical results to the base model.

Image Transformer (1/2)

Table 4. Bits/dim on CIFAR-10 test and ImageNet validation sets. The Image Transformer outperforms all models and matches PixelCNN++, achieving a new state-of-the-art on ImageNet. Increasing memory block size (*bsize*) significantly improves performance.

Model Type	*bsize*	NLL	
		CIFAR-10 (Test)	ImageNet (Validation)
Pixel CNN	-	3.14	-
Row Pixel RNN	-	3.00	3.86
Gated Pixel CNN	-	3.03	3.83
Pixel CNN++	-	2.92	-
PixelSNAIL	-	**2.85**	3.80
Ours 1D local (8l, cat)	8	4.06	-
	16	3.47	-
	64	3.13	-
	256	2.99	-
Ours 1D local (cat)	256	2.90	**3.77**
Ours 1D local (dmol)	256	2.90	-

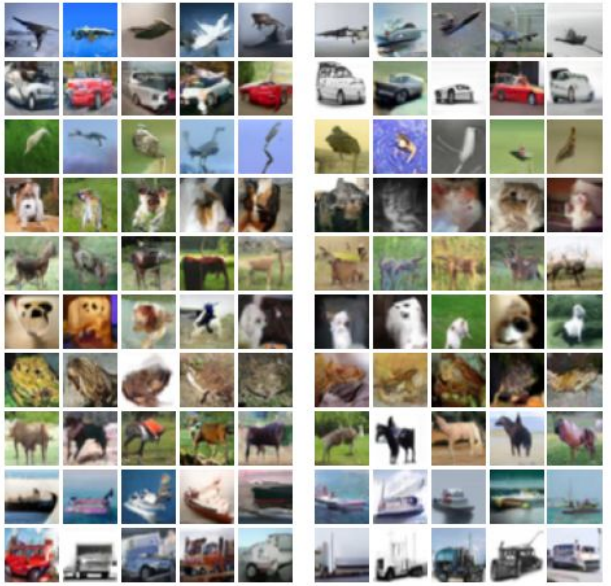


Table 3. Conditional image generations for all CIFAR-10 categories. Images on the left are from a model that achieves 3.03 bits/dim on the test set. Images on the right are from our best non-averaged model with 2.99 bits/dim. Both models are able to generate convincing cars, trucks, and ships. Generated horses, planes, and birds also look reasonable.

Image Transformer (2/2)

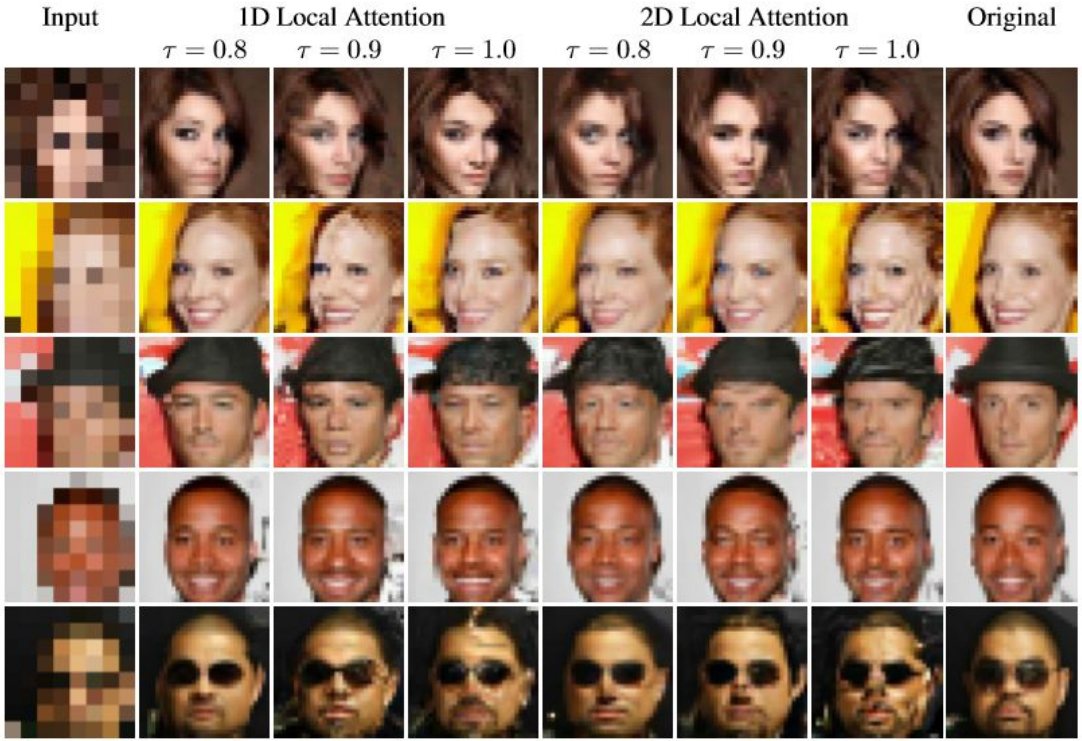


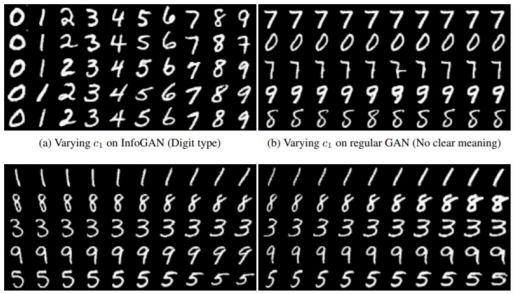
Table 6. Images from our 1D and 2D local attention super-resolution models trained on CelebA, sampled with different temperatures. 2D local attention with $\tau = 0.9$ scored highest in our human evaluation study.

INFOGAN (1/2)

Lemma 5.1 *For random variables X, Y and function $f(x, y)$ under suitable regularity conditions:* $\mathbb{E}_{x\sim X, y\sim Y|x}[f(x, y)] = \mathbb{E}_{x\sim X, y\sim Y|x, x'\sim X|y}[f(x', y)]$.

By using Lemma 5.1, we can define a variational lower bound, $L_I(G, Q)$, of the mutual information, $I(c; G(z, c))$:

$$\begin{aligned} L_I(G, Q) &= E_{c\sim P(c), x\sim G(z,c)}[\log Q(c|x)] + H(c) \\ &= E_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \\ &\leq I(c; G(z, c)) \end{aligned} \tag{5}$$



(a) Varying c_1 on InfoGAN (Digit type) (b) Varying c_1 on regular GAN (No clear meaning)

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation) (d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Figure 2: **Manipulating latent codes on MNIST:** *In all figures of latent code manipulation, we will use the convention that in each one latent code varies from left to right while the other latent codes and noise are fixed. The different rows correspond to different random samples of fixed latent codes and noise. For instance, in (a), one column contains five samples from the same category in c_1, and a row shows the generated images for 10 possible categories in c_1 with other noise fixed.* In (a), each category in c_1 largely corresponds to one digit type; in (b), varying c_1 on a GAN trained without information regularization results in non-interpretable variations; in (c), a small value of c_2 denotes left leaning digit whereas a high value corresponds to right leaning digit; in (d), c_3 smoothly controls the width. We reorder (a) for visualization purpose, as the categorical code is inherently unordered.

INFOGAN (2/2)

(a) Azimuth (pose) (b) Elevation

(c) Lighting (d) Wide or Narrow

Figure 3: **Manipulating latent codes on 3D Faces:** We show the effect of the learned continuous latent factors on the outputs as their values vary from -1 to 1. In (a), we show that a continuous latent code consistently captures the azimuth of the face across different shapes; in (b), the continuous code captures elevation; in (c), the continuous code captures the orientation of lighting; and in (d), the continuous code learns to interpolate between wide and narrow faces while preserving other visual features. For each factor, we present the representation that most resembles prior results [7] out of 5 random runs to provide direct comparison.

(a) Rotation (b) Width

Figure 4: **Manipulating latent codes on 3D Chairs:** In (a), the continuous code captures the pose of the chair while preserving its shape, although the learned pose mapping varies across different types; in (b), the continuous code can alternatively learn to capture the widths of different chair types, and smoothly interpolate between them. For each factor, we present the representation that most resembles prior results [7] out of 5 random runs to provide direct comparison.

(a) Continuous variation: Lighting (b) Discrete variation: Plate Context

Figure 5: **Manipulating latent codes on SVHN**: In (a), we show that one of the continuous codes captures variation in lighting even though in the dataset each digit is only present with one lighting condition; In (b), one of the categorical codes is shown to control the context of central digit: for example in the 2nd column, a digit 9 is (partially) present on the right whereas in 3rd column, a digit 0 is present, which indicates that InfoGAN has learned to separate central digit from its context.

(a) Azimuth (pose) (b) Presence or absence of glasses

(c) Hair style (d) Emotion

Figure 6: **Manipulating latent codes on CelebA:** (a) shows that a categorical code can capture the azimuth of face by discretizing this variation of continuous nature; in (b) a subset of the categorical code is devoted to signal the presence of glasses; (c) shows variation in hair style, roughly ordered from less hair to more hair; (d) shows change in emotion, roughly ordered from stern to happy.