UNIVERSITY OF MINNESOTA
Driven to Discover[SM]

EE 5561: Image Processing and Applications

Lecture 23

Mehmet Akçakaya

Recap of Last Lectures

- Classification tasks & CNN Architectures
  - LeNet, AlexNet, VGG, ResNet, DenseNet
- Visualization of feature maps of CNNs
- Segmentation & U-Net
- Regression Tasks
  - Denoising, superresolution, restoration/computational imaging
  - CNN architectures for these tasks
  - Some discussion on the state-of-the-art
- PyTorch Overview

Beyond CNNs

- Next two lectures:
  - Attention
  - Transformers
  - Early work comes from natural language processing (NLP)
  - We'll focus on image processing/computer vision applications

Attention

- What's visual attention?
- Focus on a region with "high resolution" e.g. trees
- Perceive the surrounding image in "low resolution"/blur
- We can adjust the focal point/do inference as needed, e.g. focus on cars

- How can a machine do this?

Attention

- In DL literature: Mechanism by which a NN can weight features by level of importance to a task + use this weighting to perform a task
- The latter part is important → This is not the same as understanding salient features *post*-training
- Popularity emerged from NLP
  - e.g. Sentences are structured differently in different languages, they are of arbitrary length, dependencies range beyond last seen element
  - Need to learn dependencies in a flexible manner

Attention

- Same kind of dependencies exist in image processing
- We saw this in the statistical image processing module!
  - Non-local means, BM3D, etc
- Just like in those methods, we want to learn these dependencies
  - Need to go beyond the small receptive field of a convolution
- We'll look at how this can be done

Recurrent Models of Visual Attention

- First work: Recurrent Models of Visual Attention (2014)
- Idea: CNNs use sliding windows across whole image, but humans only process areas of image most relevant for task
- Most tasks are sequential in nature → Glimpse at parts of image to achieve that task

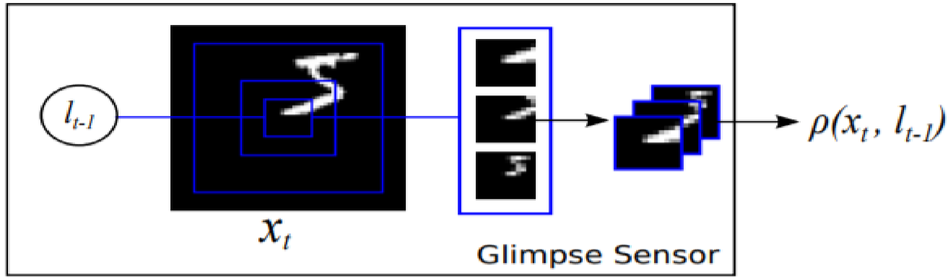
- e.g. To finding objects, humans scan the room in

such glimpses (actual recorded pattern)

Mnih et al, 2014, https://papers.nips.cc/paper/2014/file/09c6c3783b4a70054da74f2538ed47c6-Paper.pdf

Glimpse Sensor

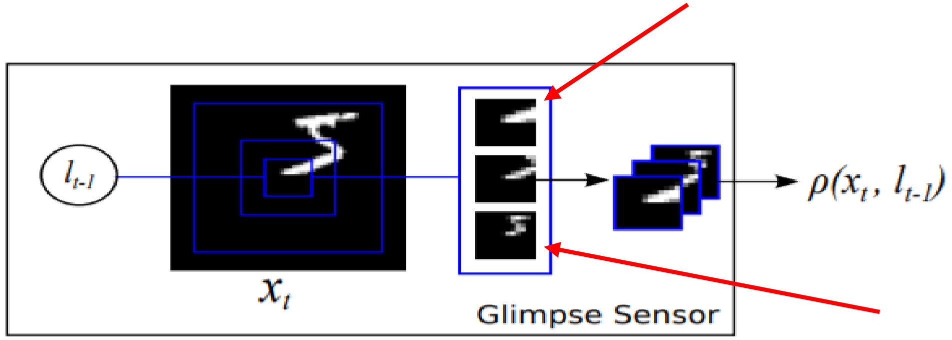
- This paper expands the idea to arbitrary tasks (using reinforcement learning along the way)
- The attention part is based on a “glimpse sensor”
- Takes in the input image and a location on that image → outputs “retina-like” representation
- Take a “glance” at the image, extract & resize the glimpse into various scales of image crops



- Each scale has the same “bandwidth”, e.g. 12×12 here

Mnih et al, 2014, https://papers.nips.cc/paper/2014/file/09c6c3783b4a70054da74f2538ed47c6-Paper.pdf

Glimpse Sensor

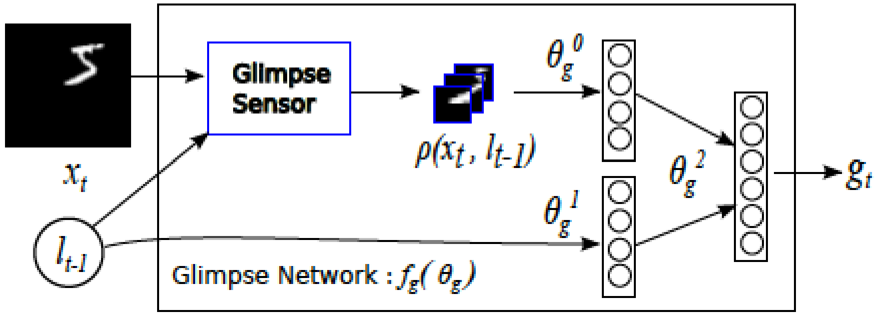


- Each scale has the same “bandwidth”, e.g. 12×12 here
- The smallest scale crop is the most detailed
- Largest crop in the outside ring is blurred
- This gives a “retina-like” representation

Mnih et al, 2014, https://papers.nips.cc/paper/2014/file/09c6c3783b4a70054da74f2538ed47c6-Paper.pdf

Glimpse Network

- The glimpse network is based on this sensor
- Takes the retina representation, flattens it
- Combines this retina representation with the glimpse location (using hidden layers + ReLU)
- Outputs single vector



- This vector contains information on “what” (retina representation) and “where” (focused location)

Mnih et al, 2014, https://papers.nips.cc/paper/2014/file/09c6c3783b4a70054da74f2538ed47c6-Paper.pdf

Glimpse Network

- This embedding is used for a specific task (e.g. object detection)
- The model is not differentiable (called “hard” attention)
- Trained using a reinforcement learning based approach

Mnih et al, 2014, https://papers.nips.cc/paper/2014/file/09c6c3783b4a70054da74f2538ed47c6-Paper.pdf

Hard vs. Soft Attention

- Hard: “Binoculars” – Seeing just part of the image (hopefully the relevant one)
  - Less computation and memory
  - Non-differentiable (as in previous paper)
- Soft: “Foggy window” – Entire image seen, but certain areas are not really attended to
  - Differentiable → Train via backpropagation as usual

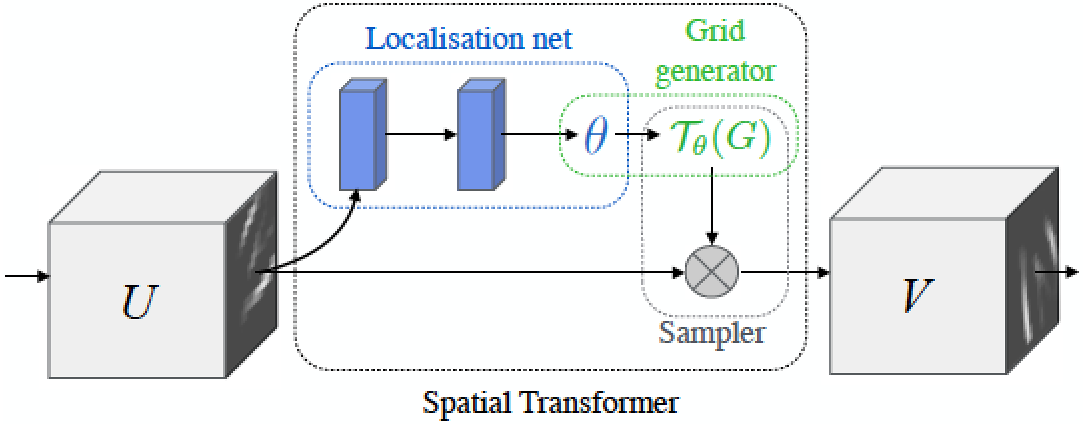
https://shairozsohail.medium.com/a-survey-of-visual-attention-mechanisms-in-deep-learning-1043eb25f343

Spatial Transformer Network

- Can we do something similar to the glimpse networks using soft attention?
- Glimpse network selectively crops portions of images as “attention”
- Instead use affine transformations
  - Handles cropping, translation, rotation, scaling and shearing
  - Fully differentiable
- Network learns the affine transformation parameters

Jaderberg et al, 2015, https://papers.nips.cc/paper/2015/hash/33ceb07bf4eeb3da587e268d663aba1a-Abstract.html

Spatial Transformer Network



- Input image, U, passed to the localisation net, which has learnable parameters θ, i.e.

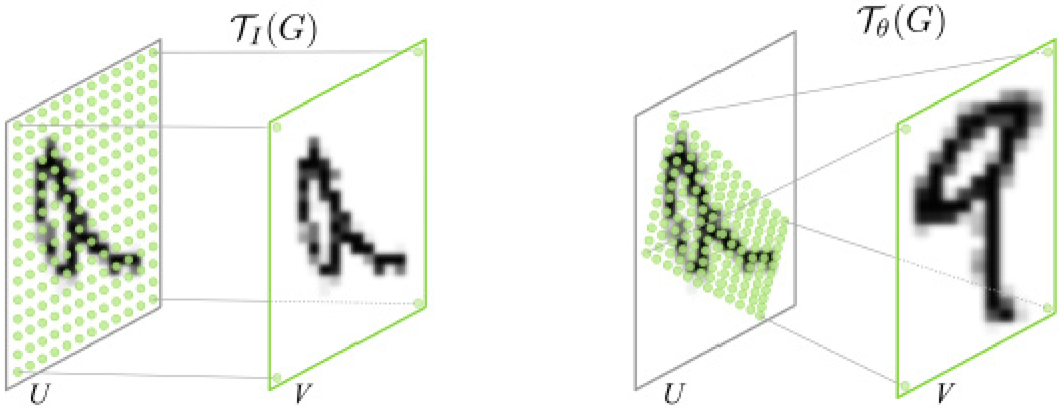
$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathtt{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

- Output is “rectified and resized” by the sampler → bilinear interpolation (differentiable)

Jaderberg et al, 2015, https://papers.nips.cc/paper/2015/hash/33ceb07bf4eeb3da587e268d663aba1a-Abstract.html

Spatial Transformer Network

- Example image being affine transformed (or “attended”)



(a) (b)

https://towardsdatascience.com/visual-attention-model-in-deep-learning-708813c2912c

More on Attention

- So far we talked about hard vs soft attention
- Now we will consider how attention can be computed
- Attention can be computed between different tensors
  - Common in machine translation – e.g. between two languages
- In image processing/computer vision, we tend to work more with attention within the image
  - Attention computed on input feature maps

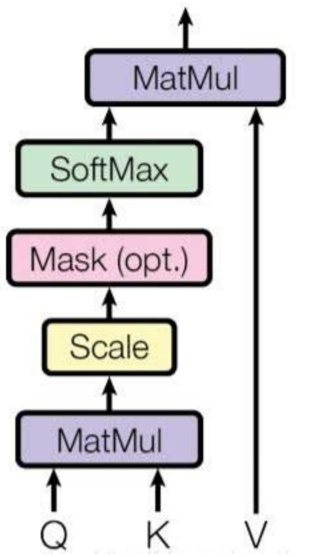
Self-Attention

- In essence, each pixel of a feature map has an associated array of attention weights for every other pixel in the map
- Useful in modelling long-term dependencies
- Gained popularity in NLP
  - Important to model long-term dependencies
  - More difficult for longer text sequences
- Also beneficial for images
- CNNs can do this with increased receptive fields, but these become inefficient with large number of parameters
- Draws inspiration from retrieval systems (e.g. search engines)
  - Map user query to keys (like title/description of website)
  - Then display matched pages/values

Self-Attention

- Self-attention does something similar
  - Input tensor of size C×H×W is mapped to three latent representations: *Query*, *key*, *value*
  - *Query* & *key* are of arbitrary hidden dimension, e.g. C’
  - Similarity between *query* & *key* measured via dot product over the channels → Tensor of size H×W×H×W
  - This contains attention weights for all combinations of input elements
  - Then multiply it with the *value* representation to get the final C×H×W attended feature maps

Scaled Dot-Product Attention

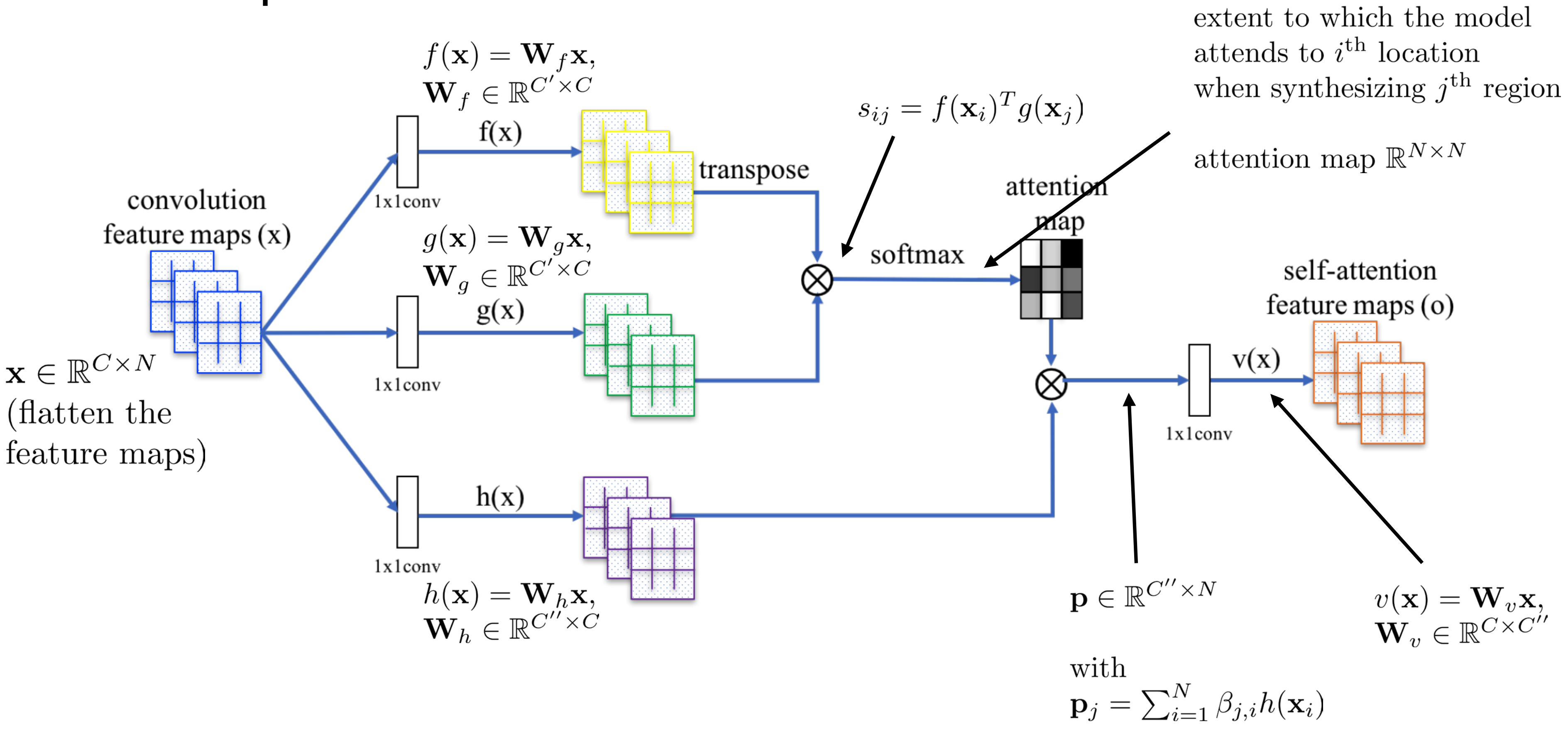


- Different from retrieval: Model decides everything based on input, i.e. no user issuing query etc

Self-Attention

- Pictorial example of a self-attention “module” [1]

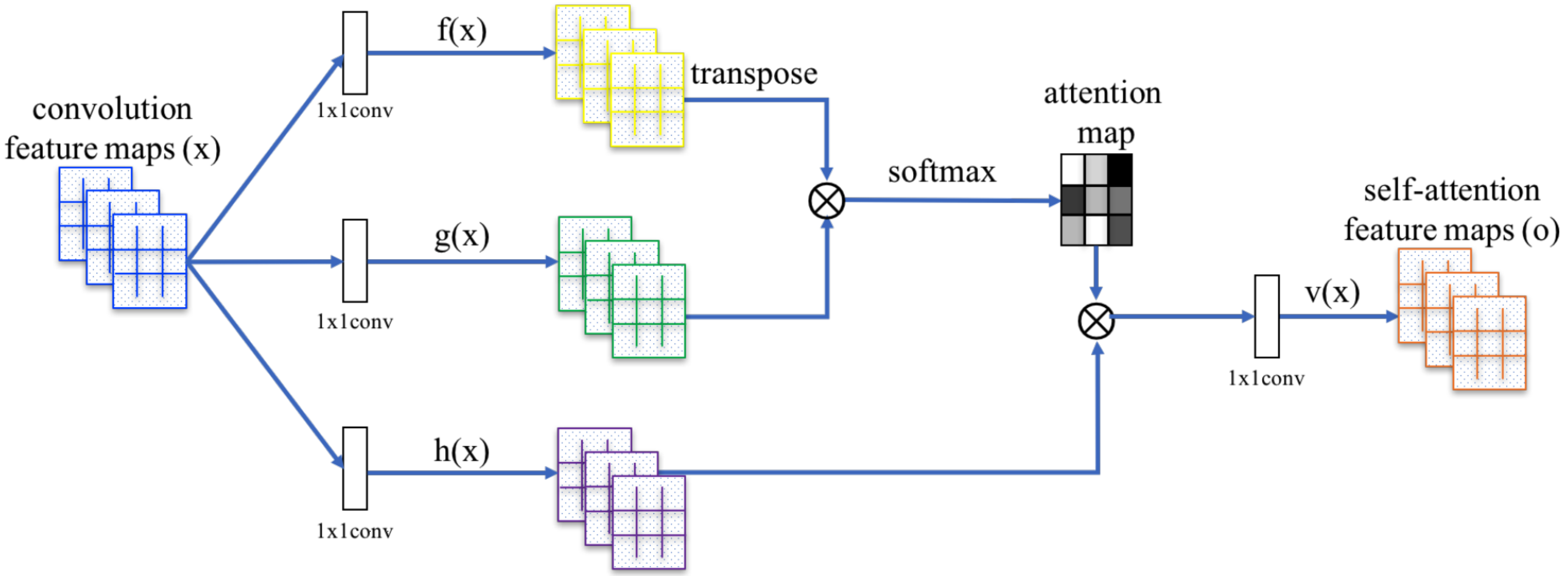
$$\beta_{j,i} = \frac{exp(s_{ij})}{\sum_{i=1}^{N} exp(s_{ij})}$$



Zhang et al, 2018, https://arxiv.org/abs/1805.08318

Self-Attention

- Pictorial example of a self-attention "module" [1]

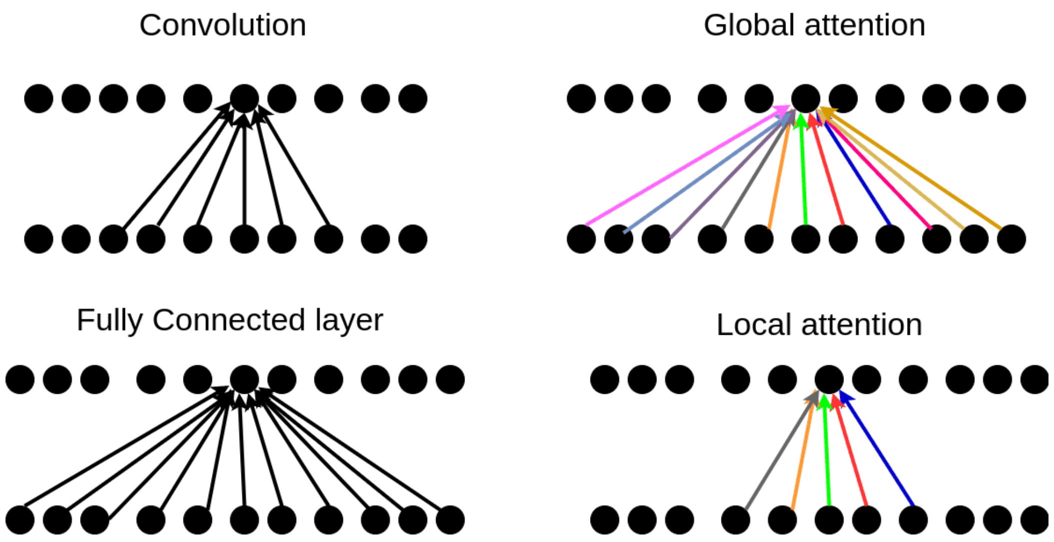


- Popularized by Vaswani et al, *Attention is all you need*, 2017.
  - Proposed a new architecture called transformer (next lecture)

Zhang et al, 2018, https://arxiv.org/abs/1805.08318

Attention Overview

- Convolutions vs. fully-connected layers vs. attention



- For attention: Weights change based on the input
- High-level: Fully-connected layer: $f(\mathbf{x}) = \mathbf{W}\mathbf{x}$

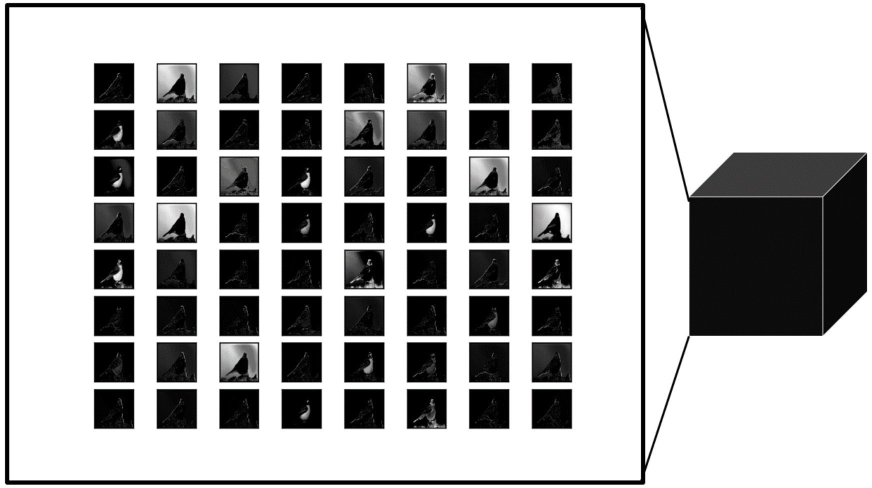
Attention: $attn(\mathbf{x}) = \mathbf{W}\mathbf{x}$

$$f(\mathbf{x}) = attn(\mathbf{x}) * \mathbf{x}$$

https://theaisummer.com/attention/

Channel and Spatial Attention

- Alternative: Determine exactly one attention weight for each element of the C×H×W input tensor
  - Layer-focused (i.e. identify which pixels each layer should focus on)
  - Contrast to self-attention, which is pixel-focused (i.e. which pixels correlate with each other)
- Spatial attention → attention "within" each feature map

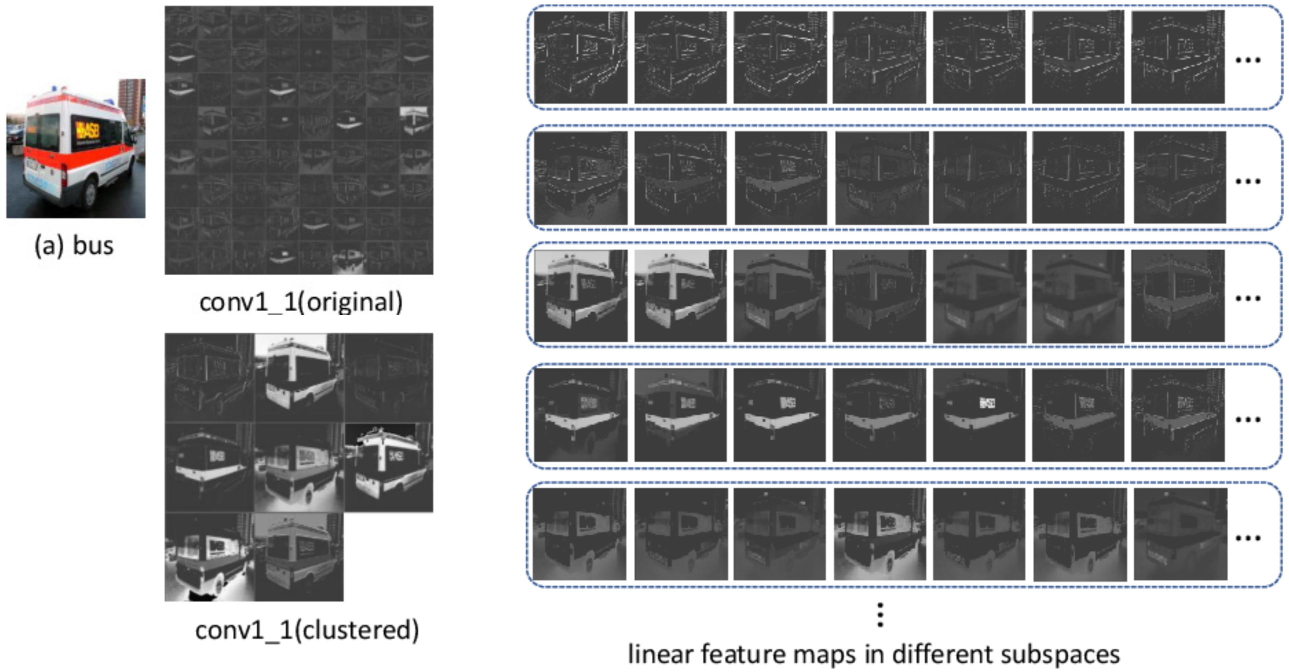


  - In this example, spatial attention will generate a mask that enhances the features of the bird

https://blog.paperspace.com/attention-mechanisms-in-computer-vision-cbam/

Channel and Spatial Attention

- Channel attention
  - Same idea across channels – Why?
  - In convolutional layers, some filters learn edges, others textures etc.
  - While feature maps look similar (~ appear like copies of each other), they learn different features (e.g. horizontal versus vertical edges, particular textures)



https://blog.paperspace.com/channel-attention-squeeze-and-excitation-networks/

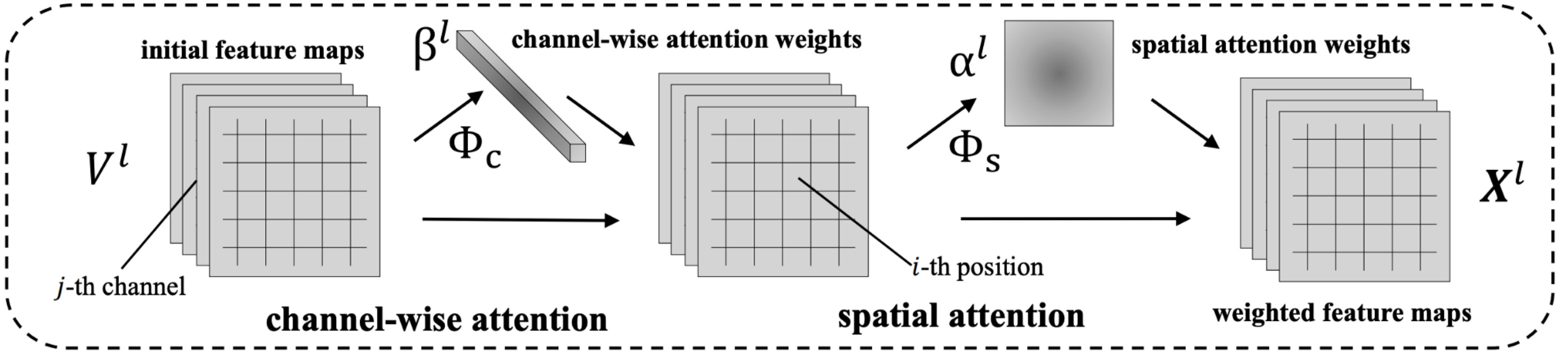
Channel and Spatial Attention

- Channel attention
  - Same idea across channels – Why?
  - In convolutional layers, some filters learn edges, others textures etc.
  - While feature maps look similar (~ appear like copies of each other), they learn different features (e.g. horizontal versus vertical edges, particular textures)
  - Channel attention → Weight each channel and enhance the channels that contribute to the overall performance
- Summary
  - Channel attention → Which feature maps are important for learning
  - Spatial attention → What *within* the feature map is important

https://blog.paperspace.com/attention-mechanisms-in-computer-vision-cbam/

Channel and Spatial Attention

- Early work: Spatial and Channel-wise Attention (SCA)-CNN
  - Applies channel attention, followed by spatial attention
  - Implemented as fully-connected layers
  - This factorization is much “cheaper” than self-attention



$\mathbf{V}^l = \mathrm{CNN}\left(\mathbf{X}^{l-1}\right)$

features

$\mathbf{h}_{t-1}$: hidden state (based on LSTM, for image captioning) – we’ll skip this part

reshape $\mathbf{V}$ to $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_C]$, $\mathbf{u}_i \in \mathbb{R}^{W \times H}$

apply average pooling to each channel: $\mathbf{v} = [v_1, v_2, ..., v_C]\,, \mathbf{v} \in \mathbb{R}^C$

$$\mathbf{b} = \tanh\left((\mathbf{W}_c \otimes \mathbf{v} + b_c) \oplus \mathbf{W}_{hc}\mathbf{h}_{t-1}\right),$$
$$\beta = \mathrm{softmax}\left(\mathbf{W}'_i\mathbf{b} + b'_i\right).$$

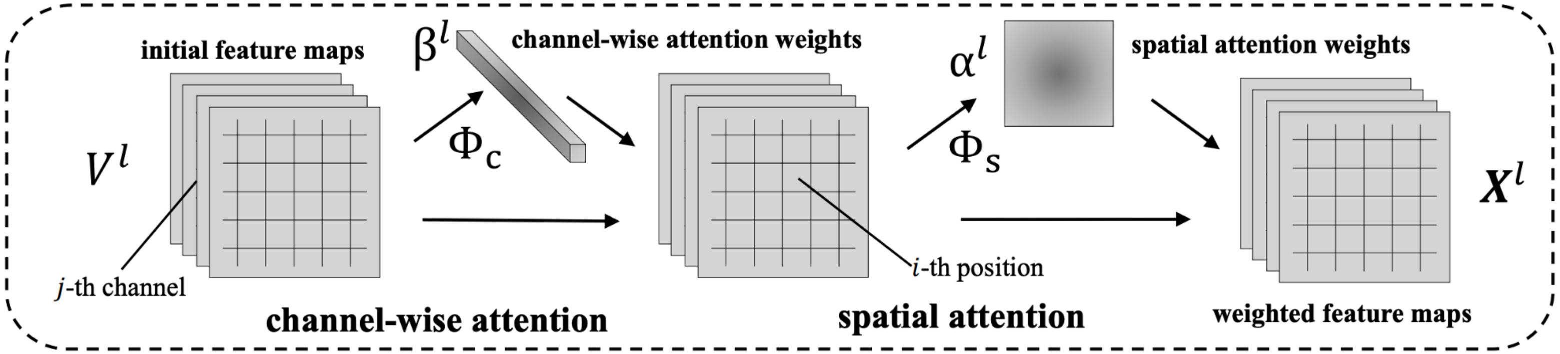
learnable weights & biases

<u>In essence:</u> $\beta = \Phi_c\left(\mathbf{h}_{t-1}, \mathbf{V}\right)$

Chen et al, 2016, https://arxiv.org/abs/1611.05594

Channel and Spatial Attention

- Early work: Spatial and Channel-wise Attention (SCA)-CNN
  - Applies channel attention, followed by spatial attention
  - Implemented as fully-connected layers
  - This factorization is much “cheaper” than self-attention



$\mathbf{V}^l = \mathrm{CNN}\left(\mathbf{X}^{l-1}\right)$

features

$\beta = \Phi_c\left(\mathbf{h}_{t-1}, \mathbf{V}\right)$

reshape $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m]$

$\mathbf{v}_i \in \mathbb{R}^C$ and $m = W \cdot H$.

$$\mathbf{a} = \tanh\left(\left(\mathbf{W}_s\mathbf{V} + b_s\right) \oplus \mathbf{W}_{hs}\mathbf{h}_{t-1}\right),$$
$$\alpha = \mathrm{softmax}\left(\mathbf{W}_i\mathbf{a} + b_i\right).$$

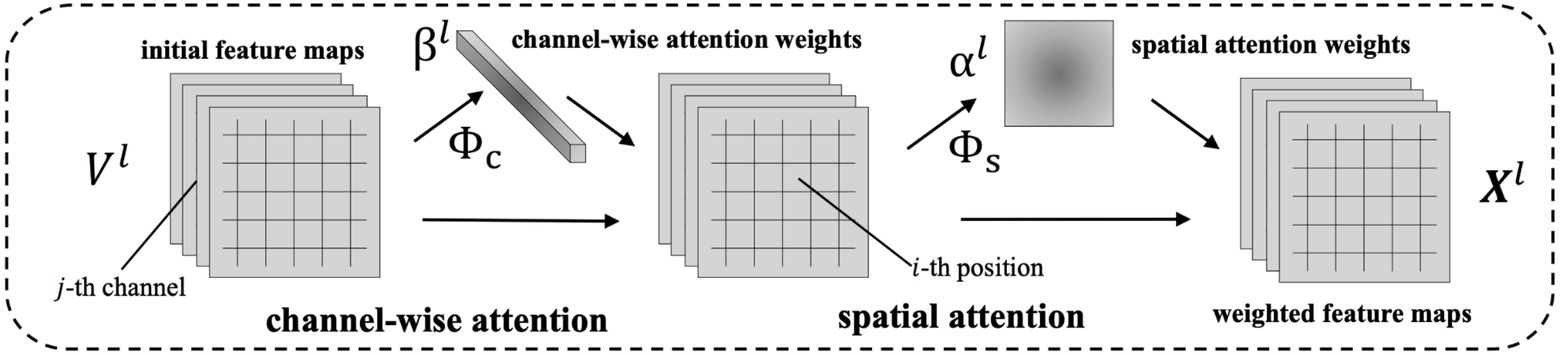
learnable weights & biases

In essence: $\alpha = \Phi_s\left(\mathbf{h}_{t-1}, f_c\left(\mathbf{V}, \beta\right)\right)$

Chen et al, 2016, https://arxiv.org/abs/1611.05594

Channel and Spatial Attention

- Early work: Spatial and Channel-wise Attention (SCA)-CNN
  - Applies channel attention, followed by spatial attention
  - Implemented as fully-connected layers
  - This factorization is much "cheaper" than self-attention



$$\mathbf{V}^l = \mathrm{CNN}\left(\mathbf{X}^{l-1}\right)$$

features

$$\beta = \Phi_c\left(\mathbf{h}_{t-1}, \mathbf{V}\right) \qquad \alpha = \Phi_s\left(\mathbf{h}_{t-1}, f_c\left(\mathbf{V}, \beta\right)\right)$$

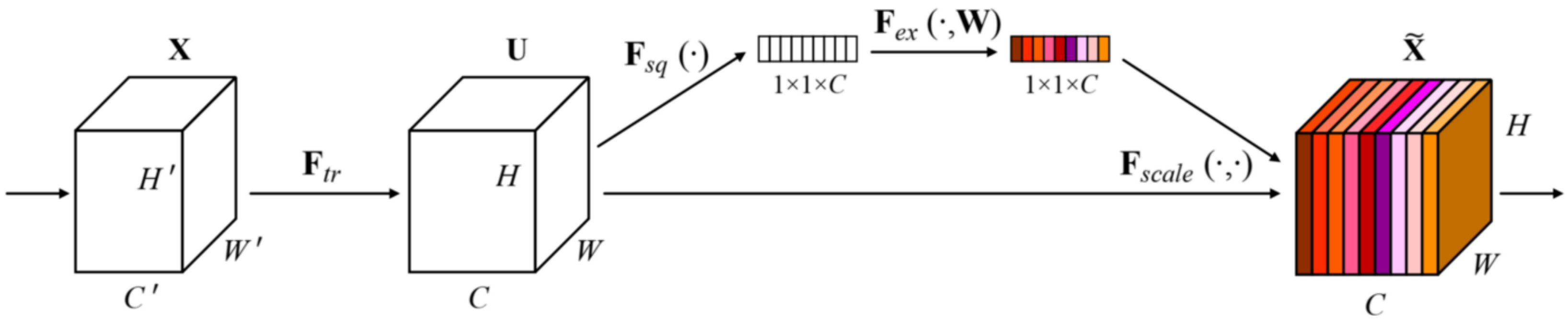
Thus:

$$\mathbf{X} = f\left(\mathbf{V}, \alpha, \beta\right).$$

Chen et al, 2016, https://arxiv.org/abs/1611.05594

Channel and Spatial Attention

- Then: Squeeze-and-Excite Network (SE-Net) (or SE block)
  - Applies global average pooling to input tensor (~SCA-CNN)
  - Called “squeeze” operation: C×H×W → C×1×1
  - Summary of information in each channel
  - This tensor is “excited” using a fully-connected bottleneck architecture



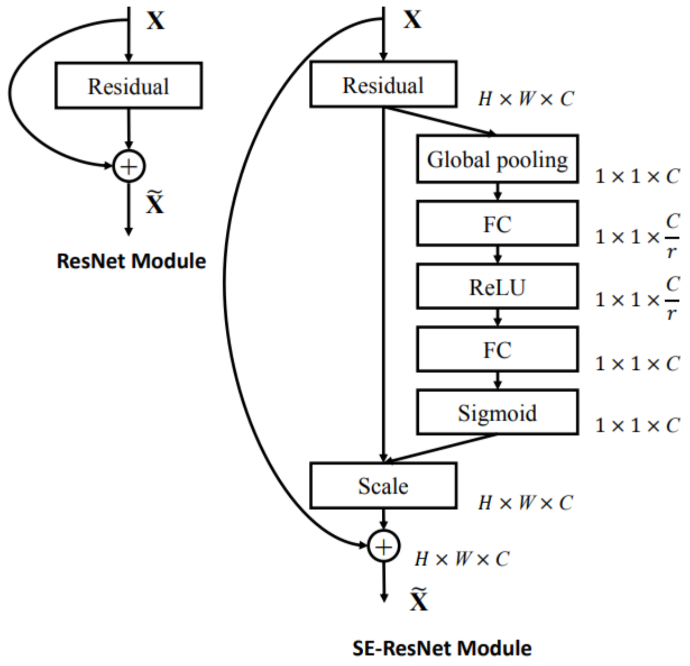
$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z})), \quad \mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C} \text{ and } \mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}.$$

  - This bottleneck leads to $2C^2/r$ additional parameter complexity
  - r: reduction ratio
  - Final output obtained by channel-wise multiplication

Hu et al, 2017, https://arxiv.org/abs/1709.01507

Channel and Spatial Attention

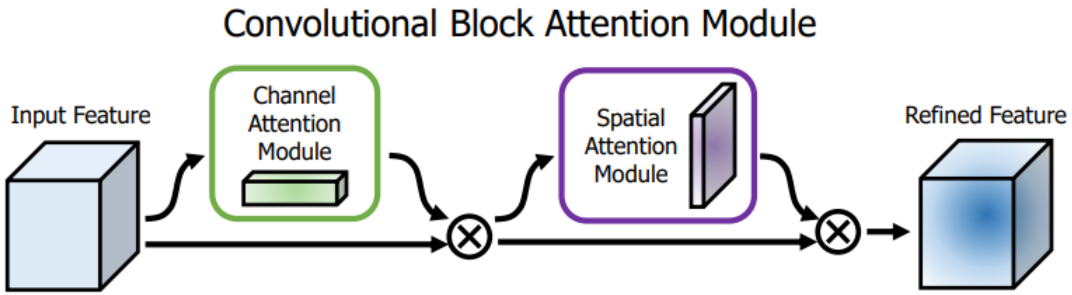
- Then: Squeeze-and-Excite Network (SE-Net) (or SE block)
  - Applies global average pooling to input tensor (~SCA-CNN)
  - Called “squeeze” operation: C×H×W → C×1×1
  - Summary of information in each channel
  - Easy to incorporate this “module” to existing architectures (~drop-in)
    - e.g. for Residual Blocks



Hu et al, 2017, https://arxiv.org/abs/1709.01507

Channel and Spatial Attention

- Then: Convolutional Block Attention Module (CBAM)
  - Mirrors SCA-CNN → channel attention followed by spatial attention

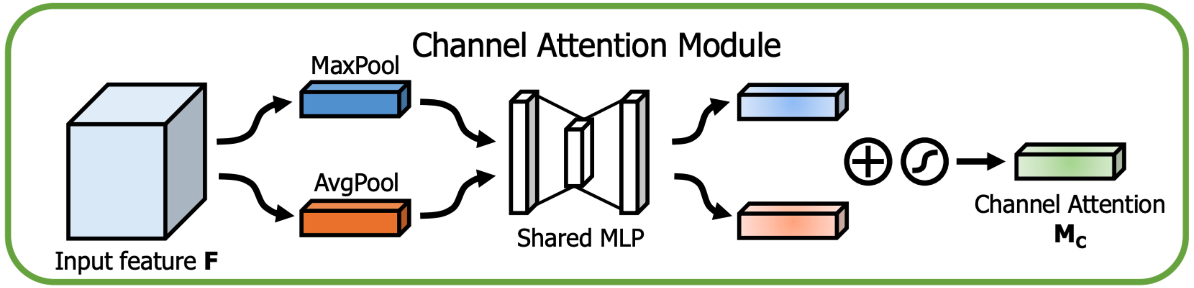


  - Difference than previous approaches: Max pooling to “summarize” across space/channels *in addition* to average pooling
  - Preserves edge features better

Woo et al, 2018, https://arxiv.org/abs/1807.06521

Channel and Spatial Attention

- Then: Convolutional Block Attention Module (CBAM)
  - Mirrors SCA-CNN → channel attention followed by spatial attention
  - Difference than previous approaches: Max pooling to “summarize” across space/channels *in addition* to average pooling
  - Channel attention module:

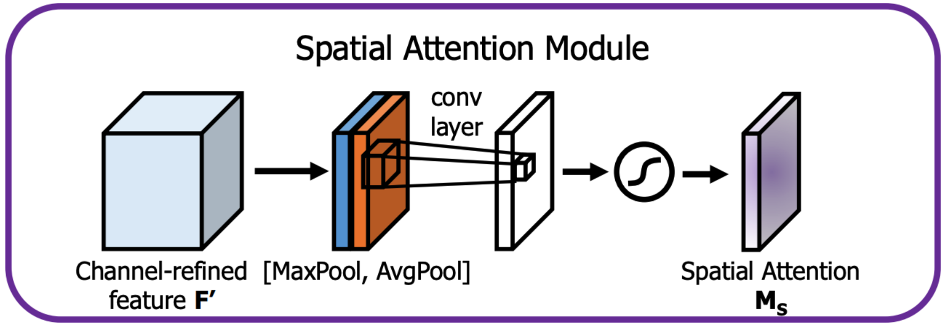


  - After the pooling, employs fully-connected bottleneck from SE-Net to derive channel attention weights

Woo et al, 2018, https://arxiv.org/abs/1807.06521

Channel and Spatial Attention

- Then: Convolutional Block Attention Module (CBAM)
  - Mirrors SCA-CNN → channel attention followed by spatial attention
  - Difference than previous approaches: Max pooling to “summarize” across space/channels *in addition* to average pooling
  - Spatial attention module:

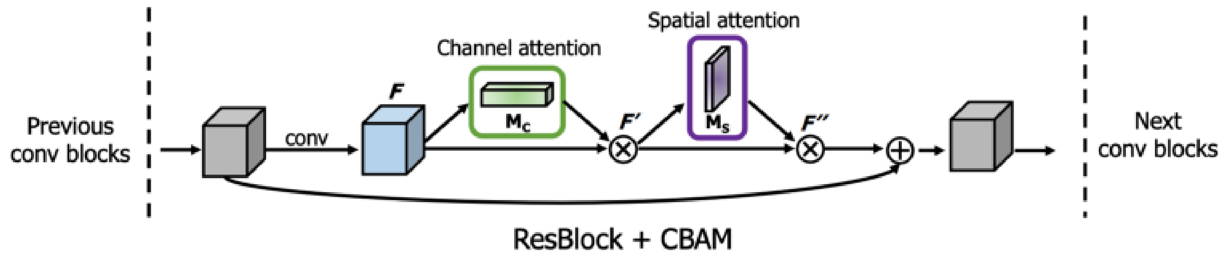


  - Again both max and average pooling for channel-pooled features
  - Then uses 2D convolutions to capture local spatial interactions
  - This mixes both max and average pooling information

Woo et al, 2018, https://arxiv.org/abs/1807.06521

Channel and Spatial Attention

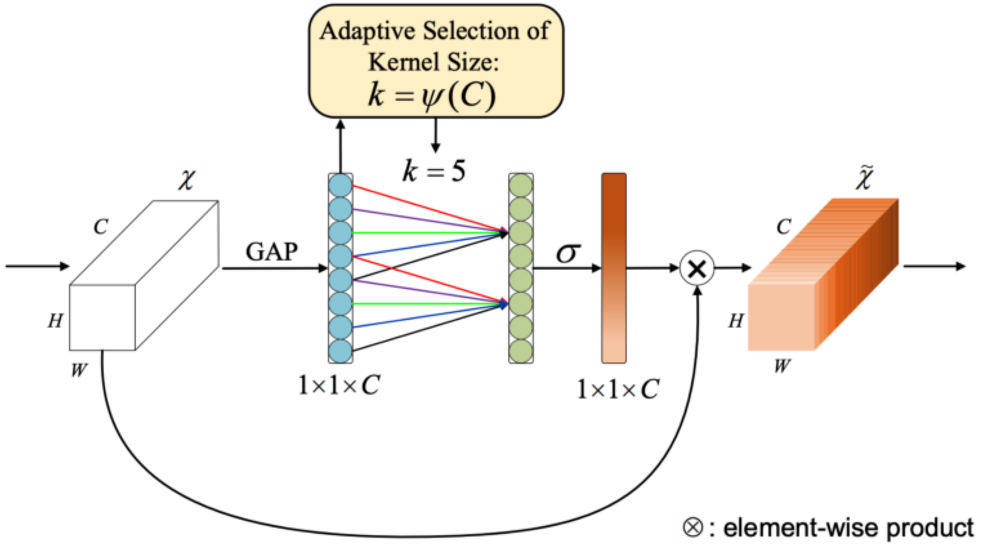
- Then: Convolutional Block Attention Module (CBAM)
  - Mirrors SCA-CNN → channel attention followed by spatial attention
  - Difference than previous approaches: Max pooling to "summarize" across space/channels *in addition* to average pooling
  - Easy to incorporate this "module" to existing architectures (~drop-in)
    - e.g. for Residual Blocks



Woo et al, 2018, https://arxiv.org/abs/1807.06521

Channel and Spatial Attention

- Is bottleneck structure the only option for attention modules?
  - No. See Efficient Channel Attention (ECA)-Net
  - Bottleneck structure may lead to inaccuracies
  - Fully-connected layer to model interactions between all channels may be inefficient
  - Replaces bottleneck with a single-layer, models channel interaction with 1D convolution



Wang et al, 2019 https://arxiv.org/abs/1910.03151

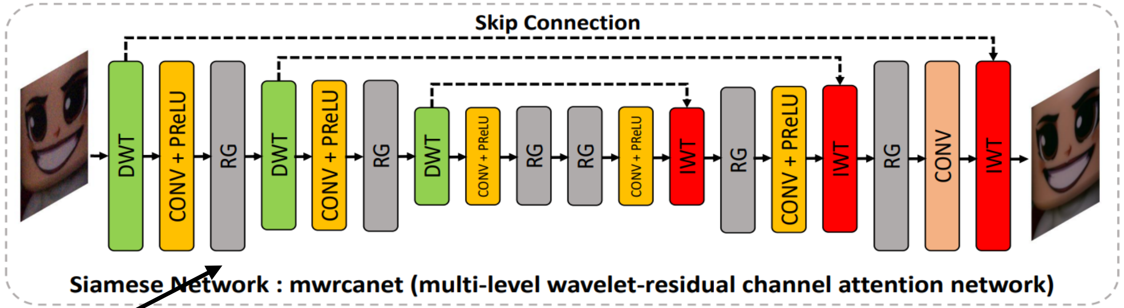
Denoising Architectures

- What about the state-of-the-art?

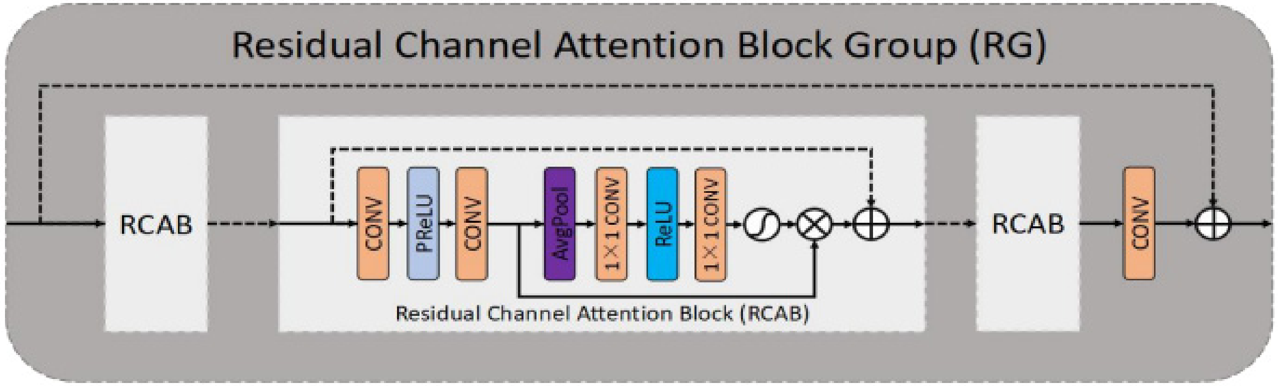
NTIRE 2020 Challenge on Real Image Denoising: Dataset, Methods and Results

https://arxiv.org/pdf/2005.04117.pdf

  - The winning architecture
    - Combines MWCNN and ResNet ideas



Special residual group (based on attention – to be covered later)



Non-local Neural Networks

- Similar idea to non-local means
- In CNNs: Long-distance dependencies are modeled by deep stacks of convolutional operations (enlarged receptive fields)
  - Inefficient computations
  - Difficult optimization (vanishing/exploding gradients)

- This paper: Make convolutional operator “global”
  - Avoid excessively deep networks
  - Improve performance (even though the operator is more computational demanding)
  - Implicitly: Uses self-attention

Wang et al, 2017, https://arxiv.org/abs/1711.07971

Non-local Neural Networks

- A non-local operation "computes the response at a position as a weighted sum of the features at all positions in the input feature maps"
- Recall classical non-local mean operation (Lecture 13):

$$\hat{\mathbf{x}}(i) = \sum_{j \in I} w(i,j)\mathbf{v}(j)$$

- Non-local operation in a deep NN is defined by

$$\mathbf{y}_i = \frac{1}{C(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j)$$

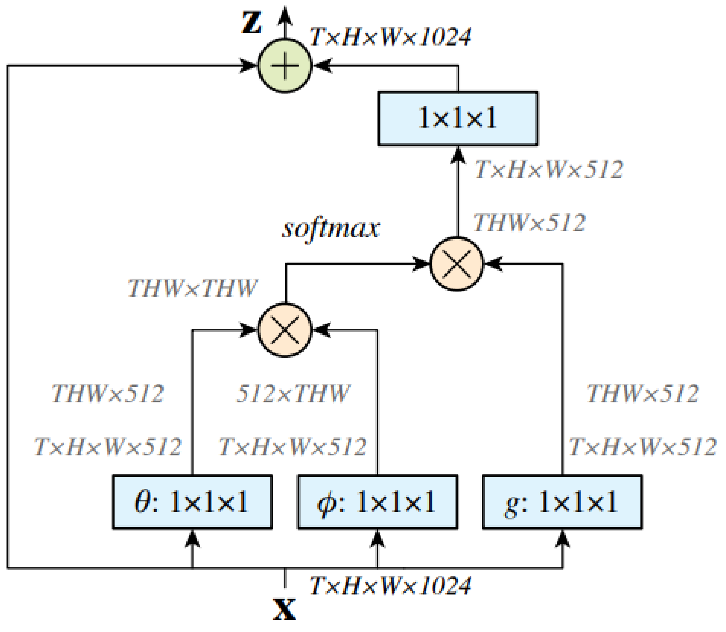
  - **x** is the input signal (image, text, video), **y** is the output signal, *i* is the position of interest and *j* enumerates all possible positions
  - Uses self-attention

Wang et al, 2017, https://arxiv.org/abs/1711.07971

Non-local Neural Networks

$$\mathbf{y}_i = \frac{1}{C(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j)$$

- $f(\cdot,\cdot)$ computes a scalar (affinity measure – similar to the Gaussian kernel in NLM)
- $g(\cdot)$ computes representation of the input signal at position j
- $C(\cdot)$ is a normalization factor
- Non-local operation, compare this to:
  - Convolutions: Sums up weighted input in local neighborhood, e.g. $i-1 \leq j \leq i+1$
  - Fully connected layers: Relationships between positions are based on learned weights – not a function of the input



Wang et al, 2017, https://arxiv.org/abs/1711.07971

Non-local Neural Networks

$$\mathbf{y}_i = \frac{1}{C(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j)$$

- *g*(·) was taken as a linear embedding, $g(\mathbf{x}_j) = \mathbf{W}_g \mathbf{x}_j$
  - $\mathbf{W}_g$ is a weight matrix that is learned (implemented as 1×1 convolution)

Wang et al, 2017, https://arxiv.org/abs/1711.07971

Non-local Neural Networks

$$\mathbf{y}_i = \frac{1}{C(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j)$$

- Multiple options were considered for *f*(·,·)
  - Gaussian

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\mathbf{x}_i^T \mathbf{x}_j}$$

  - Dot-product in embedded space

$$f(\mathbf{x}_i, \mathbf{x}_j) = \theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

  - Here θ and ϕ are also learned linear embeddings (similar to *g*)

Wang et al, 2017, https://arxiv.org/abs/1711.07971

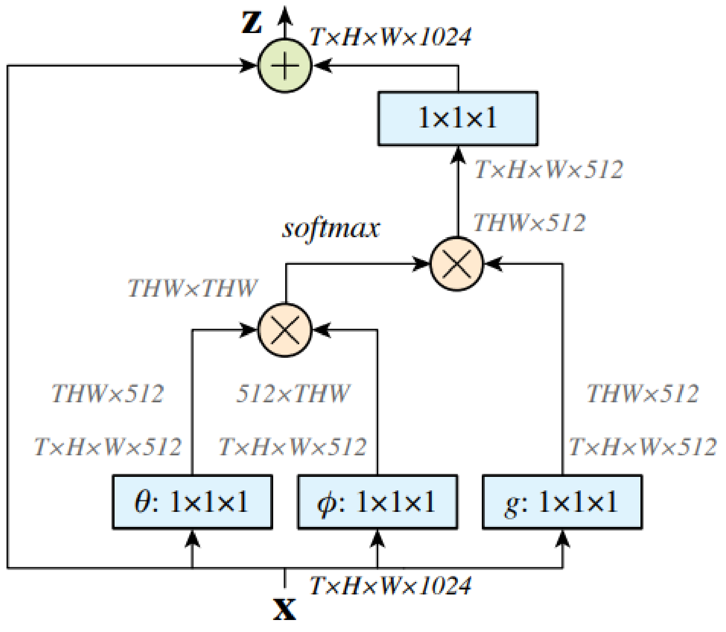
Non-local Neural Networks

$$\mathbf{y}_i = \frac{1}{C(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j)$$

- A “non-local” block implements

$$\mathbf{z}_i = \mathbf{W}_z \mathbf{y}_i + \mathbf{x}_i$$

- Note this is a residual connection (for x)
  - If the weight matrix is 0, the non-local block is identity operation
- Allows insertion into pre-trained networks
- Note matrix multiplications (& sizes) in the figure



$$\mathbf{y} = \mathrm{softmax}(\mathbf{x}^T \mathbf{W}_\theta^T \mathbf{W}_\phi \mathbf{x}) g(\mathbf{x})$$

In this (3D) example, the input has 1024 feature maps

Wang et al, 2017, https://arxiv.org/abs/1711.07971

Non-local Neural Networks

- The authors visualize the network's "attention" by finding the 20 highest weighted $\mathbf{x}_j$ for a given $\mathbf{x}_i$ position and use arrows

- Meaningful relationships for target actions in videos!

Recap

- Attention
  - Hard vs. soft attention
  - Channel vs. spatial attention
  - Self-attention
  - Non-local neural networks