Attention

Nan Ye

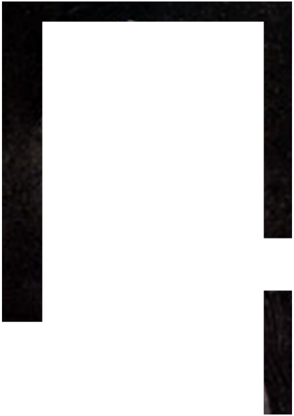
School of Mathematics and Physics
The University of Queensland

Young Woman or Old Woman?

young woman

old woman

You see what you pay attention to!



???

You see nothing if you don't pay attention!

Therefore, attention is all you need

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*, †]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*, ‡]
illia.polosukhin@gmail.com

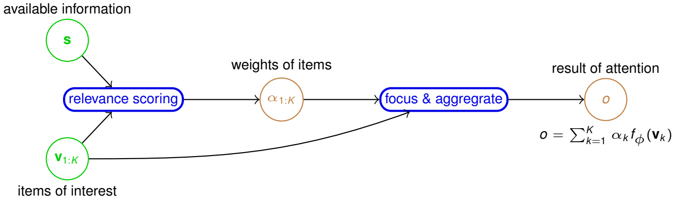
... to ~~distort~~ understand reality!

Attention Mechanism for ANNs

- Human brain receives massive amount of information at each time step, but often focuses on a tiny portion at a time, and then integrate these pieces together.
- For example, consider recognising digits in translated MNIST images.

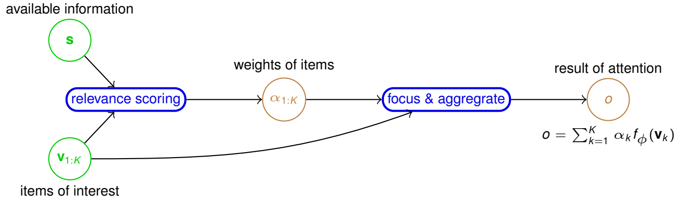
- Apparently, we do not pay much attention to the black region, but quickly focus on the white region to identify the digits.
- Such attention mechanism can be implemented in artificial neural nets as well.

A general attention framework



- Intuitively, attention is about deciding how relevant a number of items of interest are given what we already know, and then aggregating the information from the relevant items.
- What we have
  - current state $\mathbf{s}$ representing available information
  - a list of items of interest $\mathbf{v}_{1:K}$
- What we want
  - an attentive view of $\mathbf{v}_{1:K}$ given $\mathbf{s}$

A general attention framework



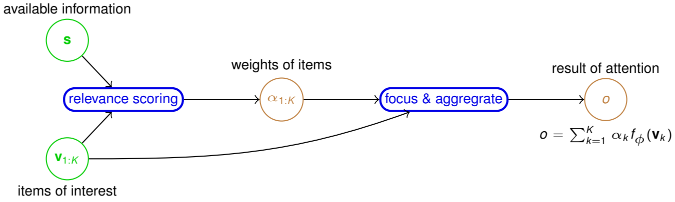
Relevance scoring

- Soft attention: maps $\mathbf{s}$ and $\mathbf{v}_{1:K}$ to a distribution $\alpha_{1:K}$.
    - implemented using a compatibility/relevance/score function r_θ

$$r_k = r_\theta(\mathbf{s}, \mathbf{v}_k), \quad k = 1, \ldots, K,$$
$$\alpha_{1:K} = \text{softmax}(r_{1:K}).$$

- Hard attention: maps $\mathbf{s}$ and $\mathbf{v}_{1:K}$ to an index in $1, \ldots, K$.
    - can be viewed as a special case of soft attention

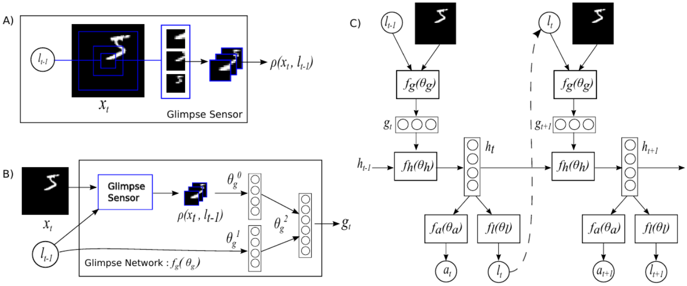
A general attention framework



Focus & aggregrate

- We can use a function f_ϕ to get a focused view on each $\mathbf{v}_k$
- The focused views are then aggregrated to $\sum_{k=1}^{K} \alpha_k f_\phi(\mathbf{v}_k)$.

Object Recognition with Attention

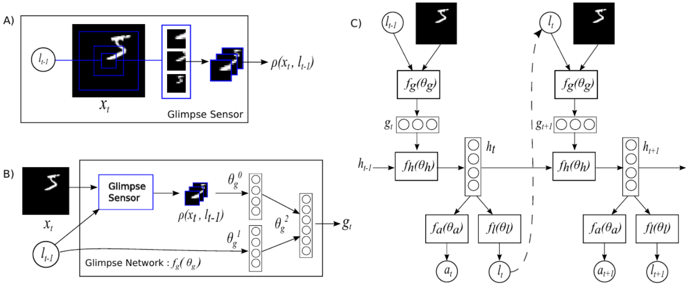


Overall architecture

- An RNN is used to make a sequence of glimpses on the image.
- At each time step
  - Glimpse network glimpses on a location to return a feature vector .
  - RNN takes in the feature vector to update its hidden state.
  - Location network proposes a new location to look at using the hidden state.
  - Action network outputs a class distribution based on current hidden state.

Mnih, Heess, Graves, et al., Recurrent models of visual attention, 2014

Object Recognition with Attention

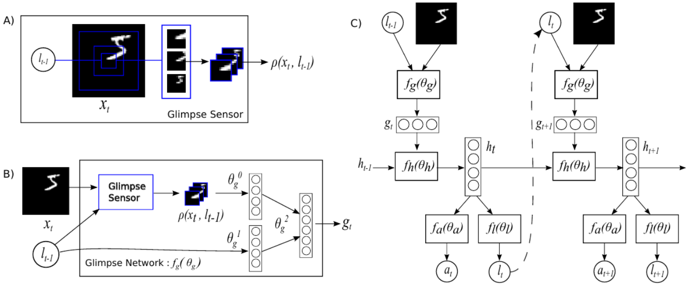


Attention inputs

- Available information: hidden state h_t of an RNN
- Items of interest: locations in an image x (together with x)

Mnih, Heess, Graves, et al., Recurrent models of visual attention, 2014

Object Recognition with Attention

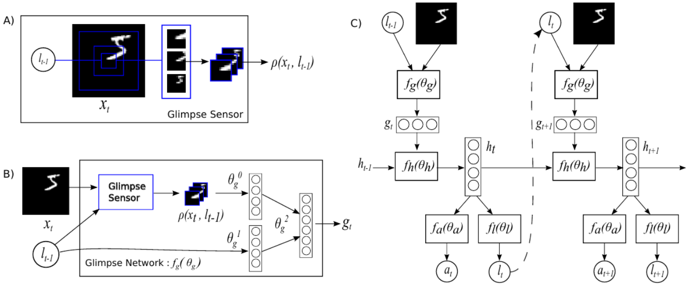


Components of attention

- Hard attention f_l: maps h_t to a location l_t at time t.
- Focused view g_t at a location l_t generated in two steps
  - (A) glimpse sensor ρ: maps x and l_t to multiple resolution patches
  - (B) glimpse network: maps $\rho(x, l_t)$ and l_t to g_t.
- RNN takes in g_t, and updates h_t to h_{t+1}

Mnih, Heess, Graves, et al., Recurrent models of visual attention, 2014

Object Recognition with Attention

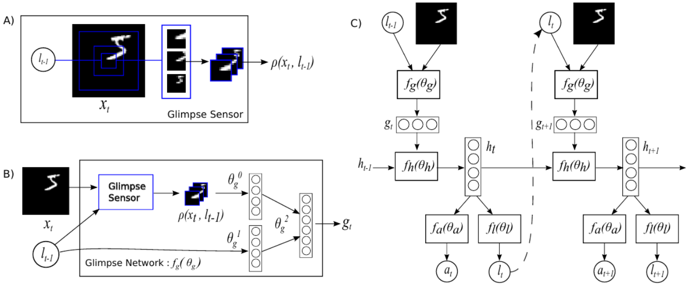


Training

- Training objective is to maximize the total classification accuracy across different time steps.
- Training algorithm is based on a reinforcement learning algorithm called REINFORCE.

Mnih, Heess, Graves, et al., Recurrent models of visual attention, 2014

Object Recognition with Attention



Prediction

- Prediction is done using the final action network.

Mnih, Heess, Graves, et al., Recurrent models of visual attention, 2014

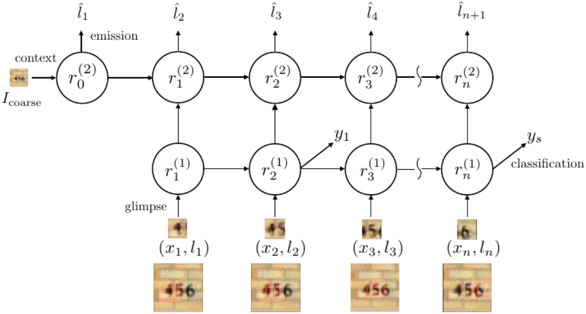
- On original MNIST dataset, RAM (recurrent attention model) is able to perform competitively as the number of glimpses increases.

Model	Error
FC, 2 layers (256 hiddens each)	1.69%
Convolutional, 2 layers	1.21%
RAM, 2 glimpses, 8×8, 1 scale	3.79%
RAM, 3 glimpses, 8×8, 1 scale	1.51%
RAM, 4 glimpses, 8×8, 1 scale	1.54%
RAM, 5 glimpses, 8×8, 1 scale	1.34%
RAM, 6 glimpses, 8×8, 1 scale	1.12%
RAM, 7 glimpses, 8×8, 1 scale	**1.07**%

- On translated MNIST dataset, RAM (recurrent attention model) is able to outperform models without an attention mechanism.

Model	Error
FC, 2 layers (64 hiddens each)	6.42%
FC, 2 layers (256 hiddens each)	2.63%
Convolutional, 2 layers	1.62%
RAM, 4 glimpses, 12×12, 3 scales	1.54%
RAM, 6 glimpses, 12×12, 3 scales	**1.22%**
RAM, 8 glimpses, 12×12, 3 scales	**1.2%**

Multiple Object Recognition with Attention



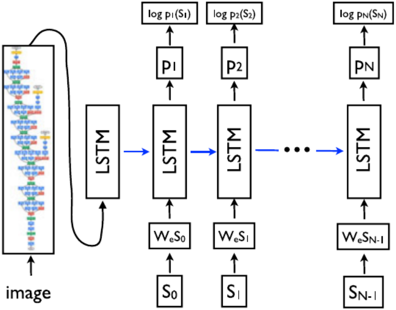
Extension of RAM to multiple object recognition

- A more sophisticated RNN architecture.
- A context network for initialising the hidden state.
- Output a label for a target after a fixed number of glimpses.
- A special end-of-sequence class is used to deal with variable number of objects.

Ba, Mnih, and Kavukcuoglu, Multiple object recognition with visual attention, 2014

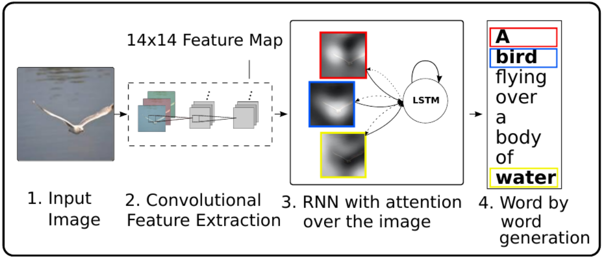
Recall: Image Captioning

- Image captioner generates a caption for a given image.
- This can be treated as a one-to-many sequence modelling problem.
- An RNN architecture



Vinyals et al., Show and tell: Lessons learned from the 2015 mscoco image captioning challenge, 2016

Image Captioning with Attention

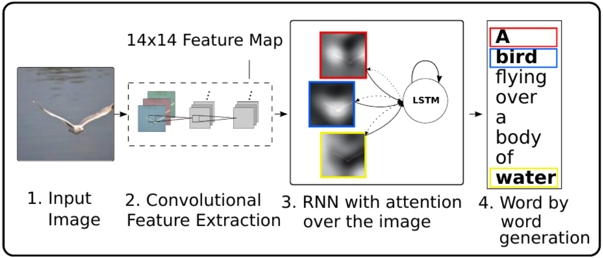


Overview

- The RNN captioner can be modified to incorporate an attention mechanism.
- Instead of using the last fully connected hidden layer of a CNN as the feature extractor, a lower layer convolutional layer can be used to provide features for parts of the image.
- The attention mechanism decides which part to look at.

Xu et al., Show, attend and tell: Neural image caption generation with visual attention, 2015

Image Captioning with Attention

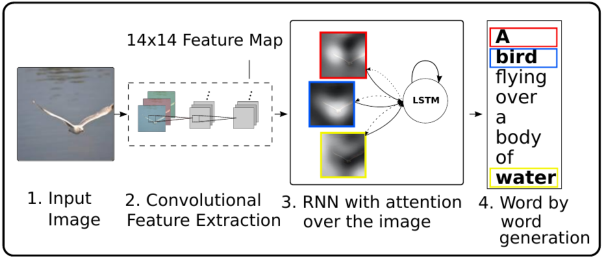


Attention inputs

- Items of interest: feature vectors $\mathbf{v}_1, \ldots, \mathbf{v}_L$ for L different locations of the image, provided by the CNN
- Available information: the hidden state $\mathbf{h}_t$, with $\mathbf{h}_0$ as the output of an MLP using $\bar{\mathbf{v}} = \frac{1}{L}\sum_i \mathbf{v}_i$ as the input.

Xu et al., Show, attend and tell: Neural image caption generation with visual attention, 2015

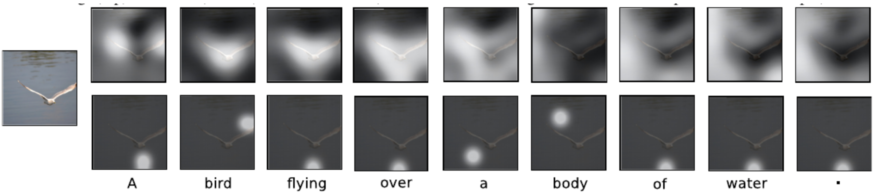
Image Captioning with Attention



Components of attention

- At time step t
  - Relevance scoring with r_θ as an MLP
    $$r_i = r_\theta(\mathbf{h}_{t-1}, \mathbf{v}_i), \qquad \alpha_{1:L} = \text{softmax}(r_{1:L})$$
  - (Aggregation) Compute $\mathbf{z}_t = \sum_{i=1}^{L} \alpha_{t,i} v_i$.
  - Compute $\mathbf{h}_t$ using $\mathbf{h}_{t-1}$, $\mathbf{z}_t$, and prev word y_{t-1}.
  - Compute d_t, distribution over vocab, using $\mathbf{h}_t, \mathbf{z}_t, y_{t-1}$
- Hard attention can be used in place of soft attention.

Xu et al., Show, attend and tell: Neural image caption generation with visual attention, 2015



Soft (top) versus hard (bottom) attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Understanding the captioner by inspecting its attention

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and a hat on a skateboard.

A person is standing on a beach with a surfboard.

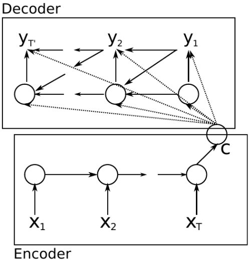
A woman is sitting at a table with a large pizza.

A man is talking on his cell phone while another man watches.

Understanding the captioner's mistakes by inspecting its attention

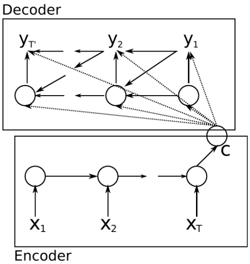
Recall: Machine Translation

- We can perform machine translation using a two-RNN architecture
  - The encoder RNN sequentially reads each word from the source sentence, and produces the final hidden state as a context vector *c* summarizing what has been seen
  - The decoder RNN produces a translation by sequentially predicting the next workd based on previous word, previous hidden state and *c*



Cho et al., Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014

Machine Translation with Attention



- Limitation: the same context vector is used for predicting all target words
- Idea: learn to attend to only the relevant source words when generating a context vector

Decoder attention inputs

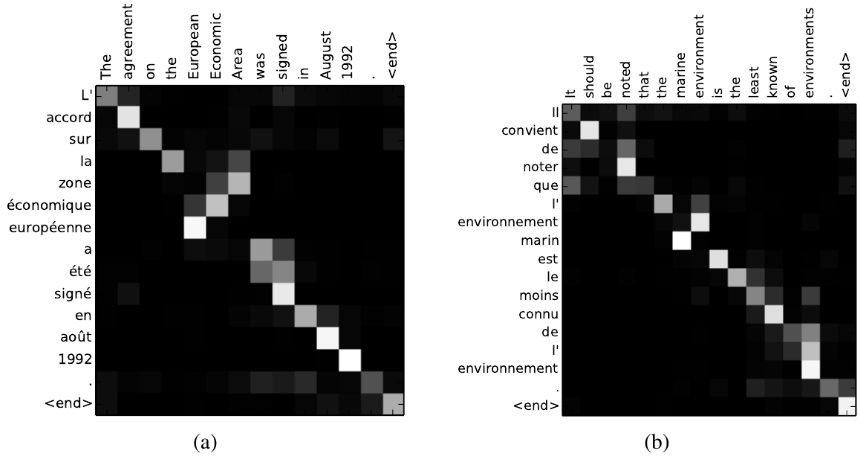
- Recall: equations for basic encoder-decoder architecture

$$\begin{aligned} \text{encoder:} \qquad & h^e_t = f^e(h^e_{t-1}, x_t), \\ \text{decoder:} \qquad & h^d_t = f^d(h^d_{t-1}, y_{t-1}, c), \\ & y_t \sim g(\cdot \mid h^d_t, y_{t-1}, c). \end{aligned}$$

  The context vector c is the final hidden state h^e_n.
- Items of interest: encoder hidden states $h^e_1, \dots, h^e_n$
- Available information: decoder hidden state h^d_t.

Components of attention

- At time step t
  - Relevance scoring with an MLP r_θ
    $$r_i = r_\theta(h^d_{t-1}, h^e_i), \qquad \alpha_{1:n} = \mathrm{softmax}(r_{1:n})$$
  - Aggregation: $c = \sum_i \alpha_i h^e_i$



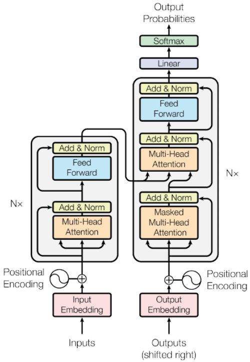
Translation from English to French

- Each row shows the attention weights over source words when generating a target word, with black and white representing 0 and 1 respectively.
- E.g. économique is generated by looking a both European and Economic, and then deciding that in French Economic comes first.

Bahdanau, Cho, and Bengio, Neural machine translation by jointly learning to align and translate, 2014

Transformer (Optional)

- The transformer is an encoder-decoder architecture for turning a sequence into another.
- It's the model of choice for many NLP problems, such as machine translation, document generation.
- It has been adapted to solve various computer vision problems.



Vaswani et al., Attention Is All You Need, 2017

- RNN-based encoder-decoder architecture:
    - uses an RNN to encode the input
    - computation at one position depends on previous positions $\Rightarrow$ hard to parallelize computation at different positions
- Transformer's encoder-decoder architecture
    - uses self-attention to compute representations of its input and output
    - self-attention allows computation at different positions to be done independently $\Rightarrow$ easy to parallelize computation at different positions
- Both types of encoder-decoder architectures allow global dependencies between input and output.

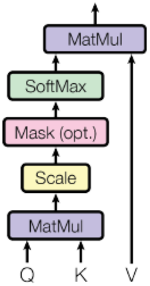
A different terminology

- Transformer's attention mechanism is described using a different terminology as compared to the general framework
- Current available information is called a query.
  - We can stack a set of queries $q_1, \ldots, q_n \in \mathbf{R}^{d_k}$ as a matrix $Q \in \mathbf{R}^{n \times d_k}$.
- Items of interest are some key-value pairs $(k_1, v_1), \ldots, (k_n, v_n) \in \mathbf{R}^{d_k} \times \mathbf{R}^{d_v}$.
  - We can stack the keys and values as matrices $K \in \mathbf{R}^{n \times d_k}$ and $V \in \mathbf{R}^{n \times d_v}$ respectively.
  - We can view v_i as a focused view for the i-th item.

Scaled dot-product attention

- Transformer uses an attention mechanism known as the scaled dot-product attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V.$$



- That is, for a query q_i, relevance scoring is done using

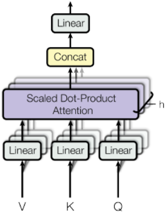
$$r_j = q_i \cdot k_j / \sqrt{d_k}, \quad j = 1, \ldots, n,$$
$$\alpha_{1:n} = \text{softmax}(r_{1:n}).$$

  The result of the attention is $\sum_j \alpha_j v_j$.

- The compability function here differs from one implemented as an MLP with q_i and k_j as the input.
- Masking can be applied to avoid attending to specific positions by setting the corresponding r_i values to $-\infty$.
- Scaled dot-product attention doesn't have any learnable parameters.

Multi-head attention

- The multi-head attention is a more general and richer class of attention mechanism than the scaled dot-product attention.
- The idea is to
  - apply h learned linear projections to the queries, keys, and values
  - apply scaled dot-product attention to each set of transformed versions of queries, keys, and values
  - concatenate the outputs of the scaled dot-product attention, and apply a learned linear transformtion to the concatenated output
- Specifically, if the queries, keys and values are all in $\mathbf{R}^d$, and we want a d-dimensional output, multi-head attention computes

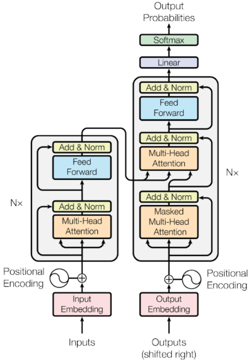


$$
\begin{aligned}
\mathrm{MultiHead}(Q, K, V) &= (z_1^\top \ldots z_h^\top) W^O, \\
\text{where } z_i &= \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V),
\end{aligned}
$$

with $W_i^Q \in \mathbf{R}^{d \times d_k}$, $W_i^K \in \mathbf{R}^{d \times d_k}$, $W_i^V \in \mathbf{R}^{d \times d_v}$, and $W^O \in \mathbf{R}^{hd_v \times d}$.

The Transformer architecture

- Input and output
  - Convert each input/output token to a d-dimensional vector using learned embeddings.
  - Add d-dimensional vector encoding positional information to the embeddings
- Encoder: $N = 6$ identical layers with 2 sub-layers
  - 1st layer: a residual multi-head self attention layer (i.e. $Q = K = V =$ outputs of previous layer) with layer normalized output.
  - 2nd layer: a residual position-wise MLP with layer normalized output
- Decoder: $N = 6$ identical layers with 3 sub-layers
  - 1 masked multi-head self attention layer, 1 encoder-decoder attention layer, 1 position-wise MLP, all with a residual connection and layer normalization
  - masking prevents attending to subsequent positions
  - during training, right-shifted output is provided as an input to the decoder
  - during prediction, the input tokens are generated by the decoder one step at a time (as in the RNN decoder)



What You Need to Know...

- Attention mechanism: a mechanism to focus on relevant input
  - a general framework
  - useful for improving performance, interpreting how a model makes prediction, and explaining model mistakes
- Applications
  - Object detection
  - Image captioning
  - Machine translation