CS 194-129 Deep Neural Networks
Spring 2018 Canny Lecture 13

1 Reading

Recurrent Models of Visual Attention

They might be better than fully connected paradigms, but convolution operations in CNNs are still very expensive (linear in the number of pixels, since we're sliding filters over the entire image). Therefore, the goal here is to extract information by adaptively selecting a sequence of regions and only processing *those* regions at high resolution.

This models the perception of humans, who focus their attention on salient parts of the visual space to gather information where and when it is required. (Think back to saccadic eye movements, and the high density of receptors in the fovea!) They then build up information from different parts of the scene, using this to guide future eye movements and decision making.

This mechanism not only saves computational resources, but reduces the task complexity as well: objects of interest can be placed in the center of the fixated region and the rest of the visual input can be ignored.

In the paper, the authors present a framework for *attention-based, task-driven visual processing with neural networks*. Their model is an RNN which processes inputs sequentially, attending to different locations one at a time and incrementally combining information to build up an internal representation of the scene. The model chooses each location based on past information and the demands of the task.

Importantly, the number of parameters and computational requirements can be controlled independently of the size of the input images. The model is trained end-to-end to maximize a performance measure which may depend on an entire sequence of decisions.

The end result? An attention-based model that can deal with clutter and large images.

The Recurrent Attention Model

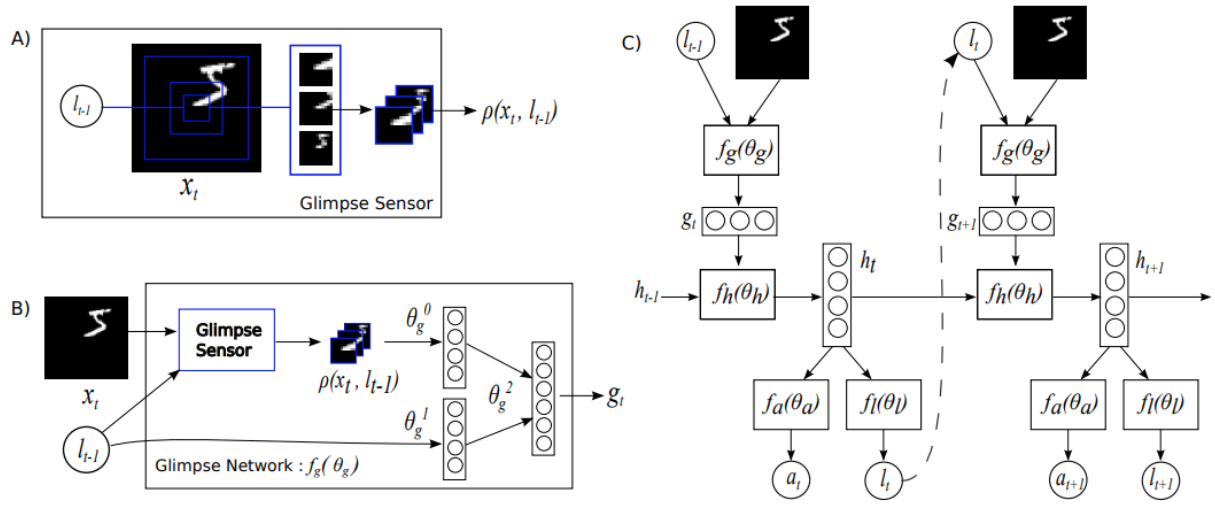
The authors consider the attention problem as "the sequential decision process of a goal-directed agent interacting with a visual environment." This agent sees a small window of the space at each time step, and is allowed to *choose the central glimpse location*. The agent can also *act in the environment*, which may in turn affect the visual input. At each step, the agent receives a scalar reward; the goal to maximize the sum of these rewards.

Therefore, at each time, the agent (i.e. our model) processes the sensor input, integrates information over time, and decides both what action to take and what location to glimpse at the next time step.

Architecture-wise, the model can be represented as the image on the next page. At each time step:

- The sensor takes an image and a location and produces a multi-resolution retina representation centered around the given location.
- This representation, along with the location, are fed through multiple linear layers to produce g_t.
- g_t (the glimpse representation) is then passed into the RNN as an input.
- The hidden output of the RNN is used as input to both a location network and an action network, which give us a location and an action, as well as to the RNN module at the next time step.

Training: the model is trained using policy gradients (specifically `REINFORCE`).



source: Recurrent Models of Visual Attention, Mnih et al.

Results

Through multiple experiments, the model has been shown to successfully combine information from multiple glimpses over time, exhibit translational invariance, search for objects in big images, and ignore clutter by centering its retina on the relevant regions.

2 Lecture

Recap

Word2vec isn't explicitly designed to do analogies (i.e. as relational offsets), but it does them pretty well. GloVe *is* designed to create embeddings that support analogical algebra, but is overall less general.

Siamese networks explicitly do comparisons of sentence embeddings, and are trained on the desired distance between paired embeddings.

Attention

Humans are very good at attending to complex stimuli (e.g. an aircraft cockpit presents an overwhelming amount of information, but pilots are trained to scan around and look for unusual, attention-requiring signs). People have then applied this idea to neural networks in order to solve problems more effectively.

There are two fundamental ways of doing attention: hard and soft attention. *Hard attention* means that the model is restricted in a binary fashion to a subset of the image: we see all of the pixels in a region and nothing else outside of that region (single location, subset of the image). Note: with a binary weight on input pixels, there's no obvious gradient that we can use to tune the attention weights. Therefore we are forced to use RL for training. We will get an overall score downstream from applying the attention model, and the gradients will propagate through the probabilities of particular pixels being in the attention map.

The attention weights in *soft attention* are continuous values between 0 to 1. We will take a weighted combination (the attention) over some inputs. Here, now, we can compute gradients and use backpropagation to train the model end-to-end.

RAM: the deep network interprets the glimpses. It needs to be aware of where it is, which is why we pass in the location as well as the retina representation. The network as a whole moves "one frame at a time."

Attention for translation: our goal is to take input in a base language and produce output in another language. This is a many-to-many sequence model, which we'll extend by giving an attention map to every LSTM unit that outputs a word. Then, for each output word, there'll be an entire distribution of weights across the input words depicting which word the current output is most related to. Unsurprisingly, the scheme works best when there's close to a one-to-one correspondence between words in each language.

RNN for captioning: we can first run our input image through a pre-trained CNN to get a semantic embedding. We then feed this embedding as the hidden state to an RNN, which starts generating an output description. Accordingly, the model only gets one look at the image.

In the attention-based version of this, the output of the CNN is still some features (with a low spatial dimension, 10×10 or something like that). We will apply an attention weight vector to these features and produce *weighted features*. Now the input to the RNN becomes both the weighted features and the current word (along with the internal state / raw features as before), and the output becomes both an attention weight vector and a distribution over words.

Assume we have from the CNN a rectangular grid of D-dimensional features $(a, b, c, ...)$. The attention map from the RNN will be a grid of the same size $(p_a, p_b, p_c, ...)$. In soft attention, we weight and add the features for each map, producing a D-dimensional context vector $z = p_a a + p_b b + p_c c....$ If the weights are continuous, the derivatives of the loss with respect to them will also be continuous. Hence everything is differentiable and we can backpropagate as normal.

If using hard attention, we will still have an attention map but we will sample from it. There'll be a 1 in one position and a 0 everywhere else, i.e. instead of weighting all the input vectors we'll just take one of them. Hence dz/dp is zero for most of the inputs and we end up having to use RL (and propagate back a probability signal instead of a simple gradient).

Attention Mechanics

Typically, soft attention has a design that involves features coming out of a feature detector, which are broadcast multiplied by an attention map and then pooled (e.g. summed) across the spatial or temporal dimensions. At the end, we have a set of attended-to features with no spatial or temporal dimensions.

(We have to broadcast multiply because there are typically extra "feature" dimensions in the features, but only spatial dimensions in an attention map.)

An attention model is trained to produce a high weight for features that have high salience. The *salience* signal (i.e. the attention gradient) is defined as the product of the activation of a layer and the gradient from that layer through to the end. "Salient" means a region has a high influence on the output, which means it has to have a high activation and a high gradient with respect to the output.

Basically, the attention model learns to reinforce things that are predictive of positive output. (In general, attention models should be liberal in what they attend to, and include anything that *might* be salient.)

Note: in a similar fashion, the i, f, and o nodes in LSTMs learn to weight features, and receive "salience gradients" during training.

As Explanation

Attention is useful in many ways. For example, it naturally allows the model to dynamically scan static inputs like images while also providing the ability to focus on relevant portions of the input. By giving the model a way to make these dependencies more explicit, attention tends to give better results on average.

More, attention offers a window into the inner workings of our networks, *and can serve as an explanation for network behavior.* (Networks are pretty cryptic otherwise, as they involve a lot of activations with no obvious meaning.)

Attention maps identify the influential parts of the input stream!

Attending to Arbitrary Regions

It can be restrictive and/or inefficient to always generate attention weights in a gridded structure. Perhaps the salient features are actually very sparse, and most of the weights end up being zero. Thus we can instead try to attend to arbitrary regions. This is what the DRAW model does.

DRAW is a generative model that looks at some images of something like digits (or other handwritten data) and then synthesizes similar outputs. It first classifies images by attending to arbitrary regions of the input, then generates images by attending to arbitrary regions of the output.

Its attention has at least three dimensions. One is the location (the center of attention). Another is the spatial extent: we let the network control the size of the window. The final dimension is the fuzziness factor: some windows can be more blurred than others.

So attention, in this scheme, is parameterized by location, scale, and blur level.

Summary

Performance. Attention models can both improve accuracy and reduce computation (usually by using hard attention, which guarantees that some parts of the input don't have to be processed at all).

Salience. Attention models learn to predict salience, i.e. to emphasize relevant input data across space or time. (The backpropagated signal through the attention map ends up signifying salience to the attention model, which then learns from that implicit signal.)

Explainability. Since attention models localize the focus of the output to a subset of the input, they serve as an explanation in themselves.

Hard vs. soft. Soft models are easier to deal with (i.e. train). Hard models, because they're not applying a differentiable weight, require RL to train. Sometimes hard models do better, so it's not that either paradigm is clearly better than the other.

References

[1] Volodymyr Mnih, Nicolas Heess, Alex Graves, Koray Kavukcuoglu. Recurrent Models of Visual Attention. In NIPS, 2014.