AI/Deep Learning State of the Art (2019)

(*well at least a sampling of the SOTA)*

Jay Urbain, PhD

References:

https://paperswithcode.com

https://www.endtoend.ai/tags/rl-weekly/

https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/

https://github.com/allenai/allennlp/blob/master/tutorials/how_to/elmo.md

https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

https://github.com/NVIDIA/vid2vid

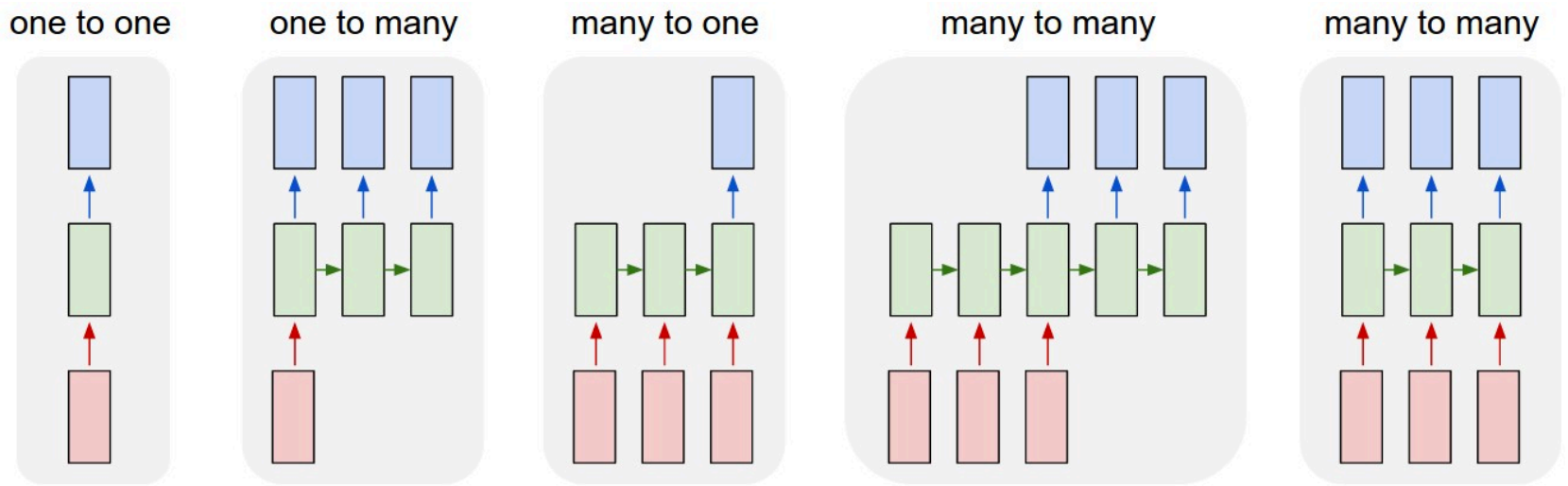
https://github.com/nashory/gans-awesome-applications

Topics

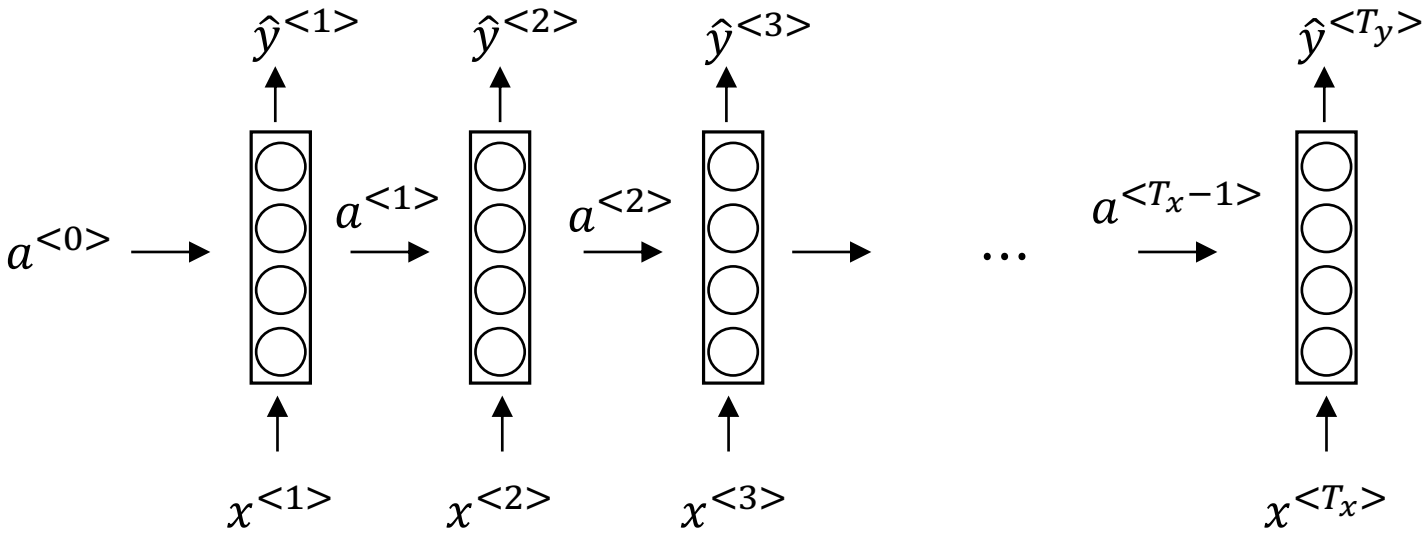
- Sequence Learning
- Transformer
- Unsupervised pretraining
- BERT
- Language Model Pretraining
- Video to Video Synthesis
- ELMo
- Modeling the structure of visual tasks
- Deep RL – Starcraft II
- Neuromorphic Architecture

Recurrent Neural Networks

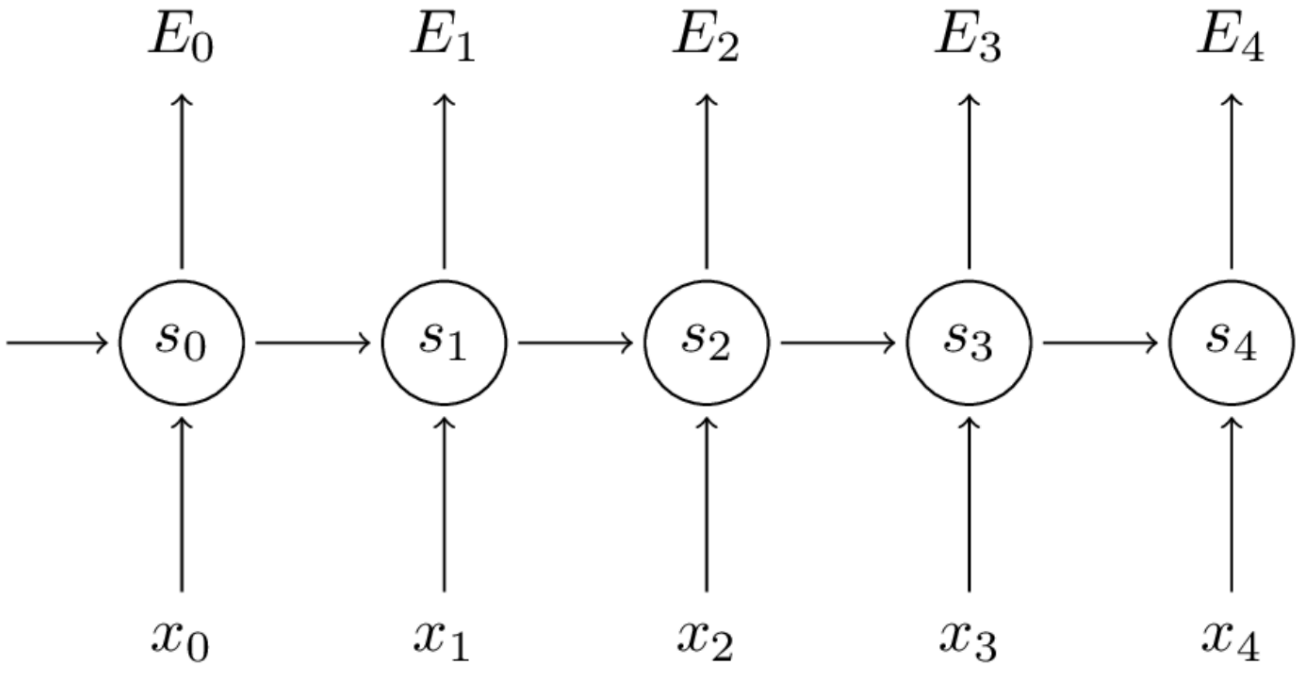
- **Sequences**. Dense Neural Networks, and CNNs only accept a fixed-sized vector as input (e.g. an image) and produce a fixed-sized vector as output (e.g. probabilities of different classes).
- RNN's can operate over Sequences in the input, the output, or in the most general case both.



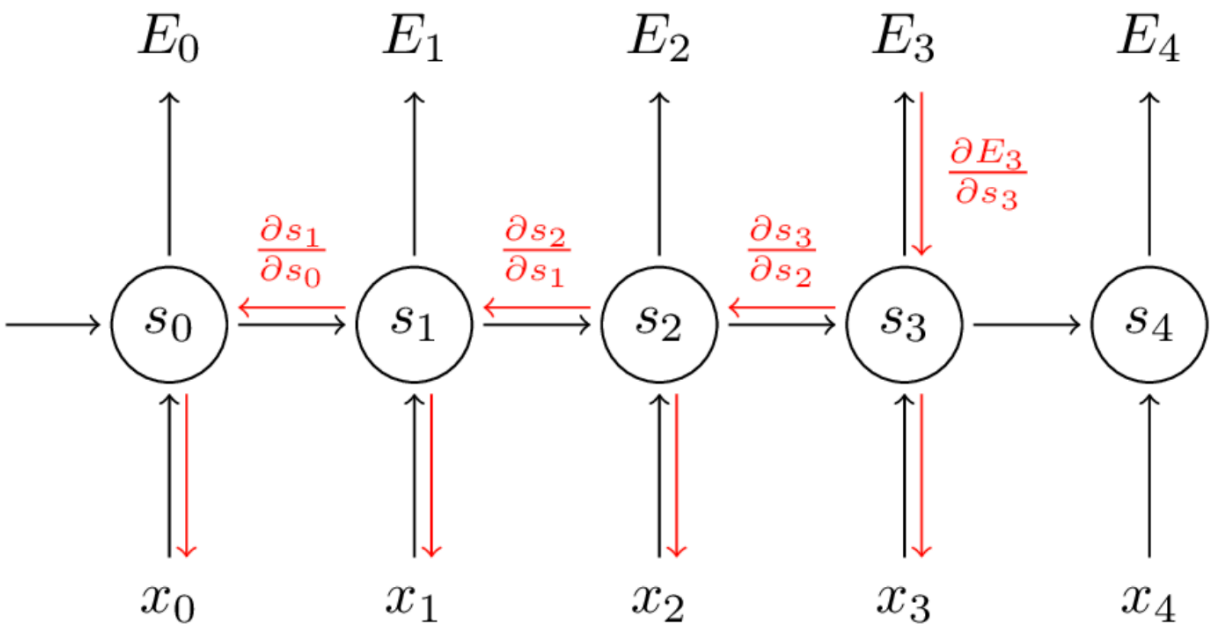
RNN's combine current input with prior state



Forward Prop



Back Prop


E_0
E_1
E_2
E_3
E_4
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
s_0
s_1
s_2
s_3
s_4
x_0
x_1
x_2
x_3
x_4

RNN forward propagation computation

- **RNN's** accept an input vector **x** and give you an output vector **y**.
- The output vector's contents are influenced by the input fed in, and the entire history of inputs you've fed in in the past.
- The RNN class has some internal state h that it gets to update every time step is called.
- This RNN's parameters are the three matrices **W_hh, W_xh, W_hy**.
- The hidden state **self.h** is initialized with the zero vector.
- There are two terms inside of the tanh:
  - one is based on the previous hidden state
  - one is based on the current input.

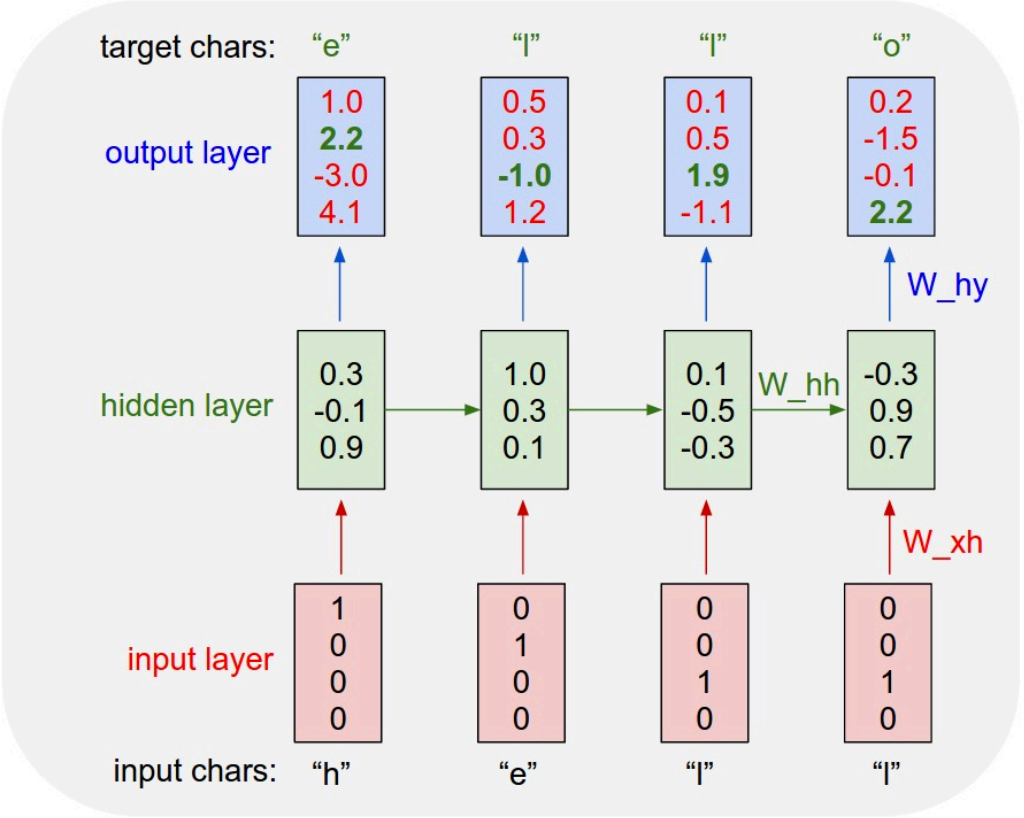
```
# forward pass
class RNN:
  # ...
  def step(self, x):
    # update the hidden state
    self.h =
      np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
    # compute the output vector
    y = np.dot(self.W_hy, self.h)
    return y

rnn = RNN()
# x is an input vector, y is the RNN's output vector
y = rnn.step(x)
```

Language models

- A language model is a generative model that can estimate the probability distribution of a set of linguistic units, typically a sequence of words.
- These are interesting models since they can be built at little cost and have significantly improved several NLP tasks such as **machine translation**, **speech recognition**, and **question/answering.**
- Historically, one of the best-known approaches is based on **Markov models** and **n-grams**.

Character-Level Language Models

- Give the RNN a huge chunk of text and ask it to *model the probability distribution* of the next character in the sequence given a sequence of previous characters.
- This will then allow the RNN to generate new text one character at a time.
- The probability of “e” should be likely given the context of “h”, 2. “l” should be likely in the context of “he”, etc.
- Each character is encoded into a vector using 1-hot-encoding encoding.
- Then observe a sequence of 4-dimensional output vectors (one dimension per character), which we interpret as the confidence the RNN currently assigns to each character coming next in the sequence.



Character-Level Language Models

- At **test time**, feed a character into the RNN and get a distribution over what characters are likely to come next.
- Sample from this distribution, and feed it right back in to get the next letter.
- Repeat this process and you're sampling text! Lets now train an RNN on different datasets and see what happens.

Sample character generation from tiny Shakespear collection -

- PANDARUS: Alas, I think he shall be come approached and the day When little srain would be attain'd into being never fed, And who is but a chain and subjects of his death, I should not sleep.
- Second Senator: They are away this miseries, produced upon my soul, Breaking and strongly should be buried, when I perish The earth and thoughts of many states.
- DUKE VINCENTIO: Well, your wit is in the care of side and that.
- Second Lord: They would be ruled after this chamber, and my fair nues begun out of the fact, to be conveyed, Whose noble souls I'll have the heart of the wars.
- Clown: Come, sir, I will make did behold your worship.
- VIOLA: I'll drink it.

notebooks/nlp/CharacterRNN

Language models – Deep Learning Evolution

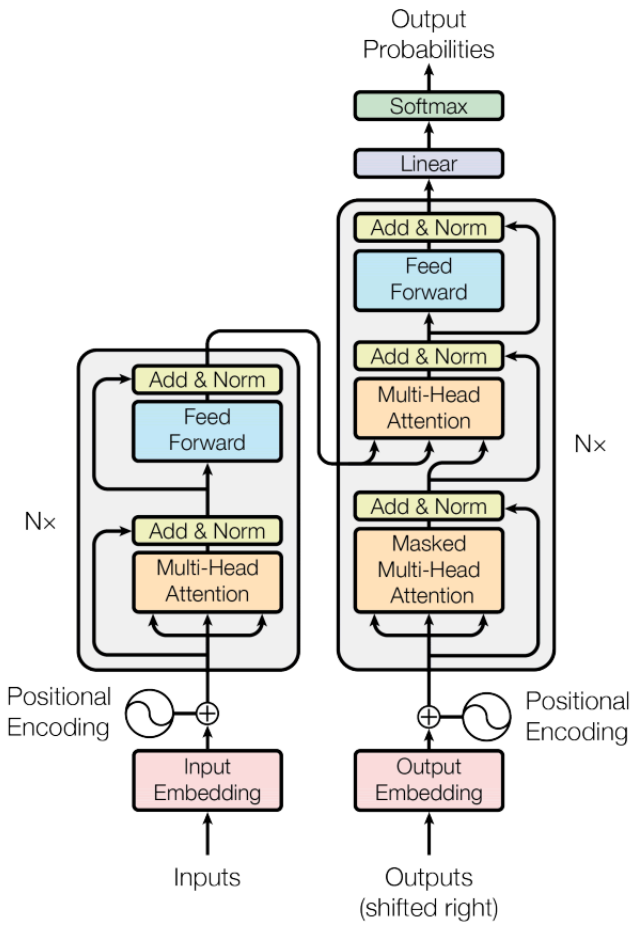
- With the emergence of deep learning, more powerful models generally based on **long short-term memory networks** (LSTM) appeared.
- Although highly effective, existing models are usually unidirectional, meaning that only the left (or right) context of a word ends up being considered, are slow to train, and finicky.
- To obtain really good performance **Attention** layers are needed to capture long-term dependencies.

Attention is all you need: Encoder-decoder Transformer model with attention, self-attention

- Propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely.
- Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train.

Example:

http://54.163.221.189:8080/



State-of-the-art: Pretraining

- One of the biggest challenges in NLP is the shortage of training data.
- Because NLP is a diversified field with many distinct tasks, most task-specific datasets contain only a few thousand or a few hundred thousand human-labeled training examples.
- Modern deep learning-based NLP models see benefits from much larger amounts of data, improving when trained on millions, or *billions*, of annotated training examples.
- To help close this gap in data, researchers have developed a variety of techniques for training general purpose language representation models using the enormous amount of unannotated text on the web.
- Known as *pre-training*.
- The pre-trained model can then be fine-tuned on small-data NLP tasks like question answering and sentiment analysis, resulting in substantial accuracy improvements compared to training on these datasets from scratch.

Language models: Google's BERT representation

- Last October, the Google AI Language team published a paper that caused a stir in the community.
- **BERT (Bidirectional Encoder Representations from Transformers)** is a new bidirectional language model that has achieved state of the art results for 11 complex NLP tasks, including **sentiment analysis**, **question answering**, and **paraphrase detection**.
- **BERT** is based on a stack of **Transformers using an encoder and decoder**

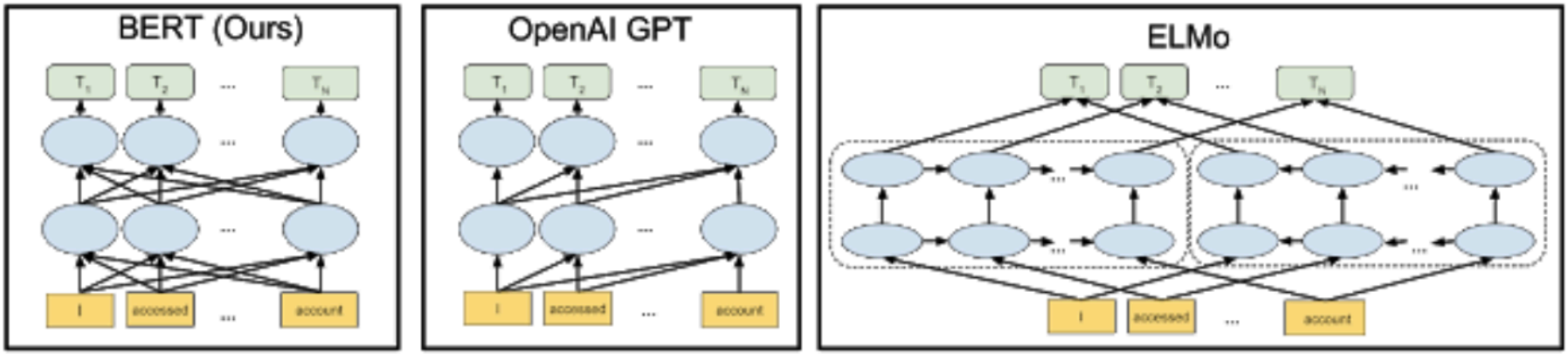
BERT: State-of-the-Art Pre-training for Natural Language Processing

- BERT builds upon recent work in pre-training contextual representations — including Semi-supervised Sequence Learning, Generative Pre-Training, ELMo, and ULMFit.
- Unlike previous models, **BERT is the first deeply bidirectional, unsupervised language representation**, pre-trained using only a plain text corpus (in this case, Wikipedia).
- Pre-trained representations can either be *context-free* or *contextual.*
- *Contextual* representations can further be *unidirectional* or *bidirectional*.

BERT: State-of-the-Art Pre-training for Natural Language Processing

- Context-free models such as **word2vec** or **GloVe** generate a single word embedding representation for each word in the vocabulary.
- Contextual models generate a representation of each word that is based on the other words in the sentence.
- For example, in the sentence "*I accessed the bank account*," a unidirectional contextual model would represent "*bank*" based on "*I accessed the*" but not "*account*."
- However, BERT represents "*bank*" using both its previous and next context — "*I accessed the ... account*" — starting from the very bottom of a deep neural network, making it deeply bidirectional.

BERT



BERT is deeply bidirectional, OpenAI GPT is unidirectional, and ELMo is shallowly bidirectional.

The Strength of Bidirectionality

- It is not possible to train bidirectional models by simply conditioning each word on its previous *and* next words.
- This would allow the word that's being predicted to indirectly "see itself" in a multi-layer model.
- To solve this problem, the authors masked out some of the words in the input and then condition each (masked) word bidirectionally to predict the masked words. For example:

Input: The man went to the $[MASK]_1$. He bought a $[MASK]_2$ of milk .
Labels: $[MASK]_1$ = store; $[MASK]_2$ = gallon

Sentence Pretraining

- BERT also learns to model relationships between sentences by pre-training on a very simple task that can be generated from any text corpus.
- Given two sentences A and B, is B the actual next sentence that comes after A in the corpus, or just a random sentence? For example:

Sentence A = `The man went to the store.`
Sentence B = `He bought a gallon of milk.`
Label = `IsNextSentence`

Sentence A = `The man went to the store.`
Sentence B = `Penguins are flightless.`
Label = `NotNextSentence`

BERT: State of the art

- Demonstrated state-of-the-art results on 11 NLP tasks, including the very competitive Stanford Question Answering Dataset (SQuAD v1.1).

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Comparative results for the GLUE Benchmark.

Video to Video Synthesis

- We are quite used to the interactive environments of simulators and video games typically created by **graphics engines**.
- While impressive, the classic approaches are costly in that the scene geometry, materials, lighting, and other parameters must be meticulously specified.

Video to Video Synthesis

Is it possible to automatically build these environments using deep learning techniques?

- In their **video-to-video synthesis paper**, researchers from NVIDIA address this problem.
- Their goal is to come up with a mapping function between a source video and a photorealistic output video that precisely depicts the input content.
- The authors model it as a distribution matching problem, where the goal is to get the conditional distribution of the automatically created videos as close as possible to that of the actual videos.
- To achieve this, they build a model based on **generative adversarial networks** (GAN).
- The key idea, within the GAN framework, is that the *generator* tries to produce realistic synthetic data such that the *discriminator* cannot differentiate between real and synthesized data.

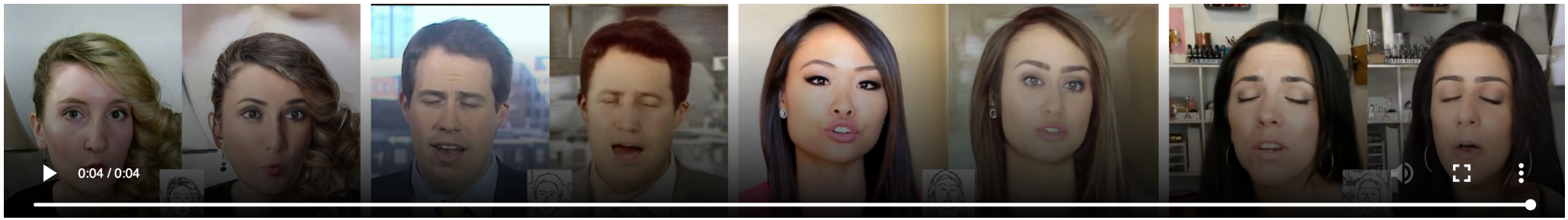
Video to Video Synthesis

- nVidia defines a spatiotemporal learning objective, with the aim of achieving temporarily coherent videos.
- The results are absolutely amazing, as can be seen in the video below.

The input video is in the top left quadrant. It is a segmentation map of a video of a street scene from the **Cityscapes dataset.** The authors compare their results (bottom right) with two baselines: pix2pixHD (top right) and COVST (bottom left).

Video to Video Synthesis

- This approach can be applied to many other tasks:
- sketch-to-video synthesis for face swapping.
- In the filmstrip below, for each person we have an original video (left), an extracted sketch (bottom-middle), and a synthesized video.



https://tcwang0509.github.io/vid2vid/paper_gifs/face.mp4

Improving word embeddings

- A **word embeddings in NLP** is a dense, vectorized representation of a word.
- Critically important in improving NLP performance.
- Similar idea to an autoencoder. Feed in word and predict context words, or feed in context and predict target word.
- Can perform semantic vector operations on embeddings, i.e. King - Man + Woman = Queen).

Improving word embeddings - ELMo

ELMo is a deep *contextualized* word representation that models:

1. complex characteristics of word use (e.g., syntax and semantics), and
2. how these uses vary across linguistic contexts (i.e., to model polysemy).

- These word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus.
- They can be easily added to existing models and significantly improve the state of the art across a broad range of challenging NLP problems, including question answering, textual entailment and sentiment analysis.

Improving word embeddings - ELMo

ELMo representations are:

- *Contextual*: The representation for each word depends on the entire context in which it is used.
- *Deep*: The word representations combine all layers of a deep pre-trained neural network.
- *Character based*: ELMo representations are purely character based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens unseen in training.

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Modeling the structure of visual tasks

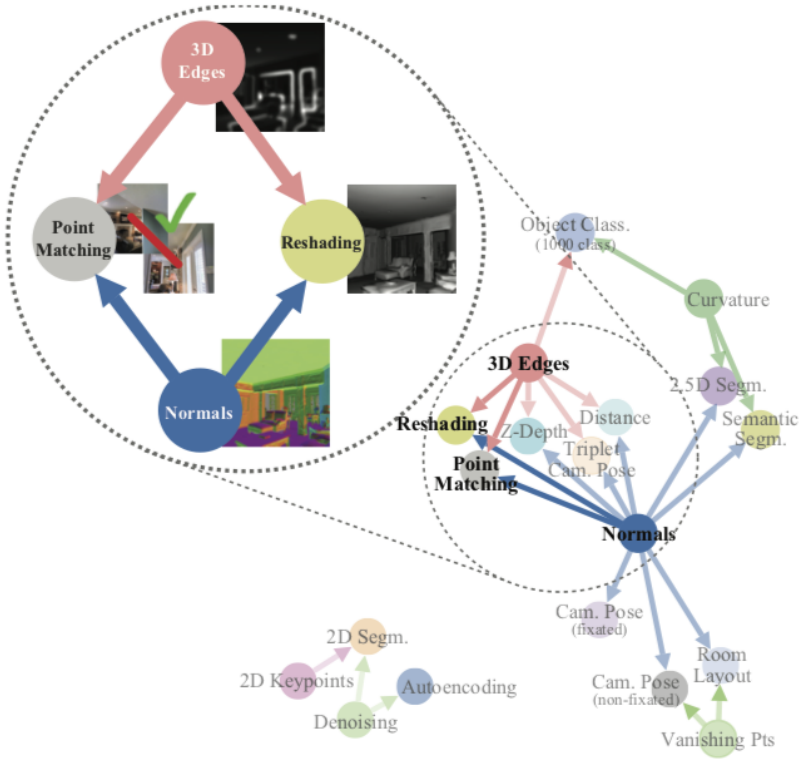
Are visual tasks related or not?

- This is the question addressed by researchers at Stanford and UC Berkeley in the paper titled, **Taskonomy: Disentangling Task Transfer Learning**, which won the **Best Paper Award** at **CVPR 2018**.
- It can be argued that some kind of connection exists between certain visual tasks. For example, knowing surface normals can help in estimating the depth of an image. In such a scenario, **transfer learning** techniques – or the possibility to reuse supervised learning results – are very useful.

Modeling the structure of visual tasks

Are visual tasks related or not?

- The authors propose a computational approach to modeling this structure by finding transfer-learning dependencies across 26 common visual tasks, including object recognition, edge detection, and depth estimation.
- The output is a computational taxonomy map for task transfer learning.



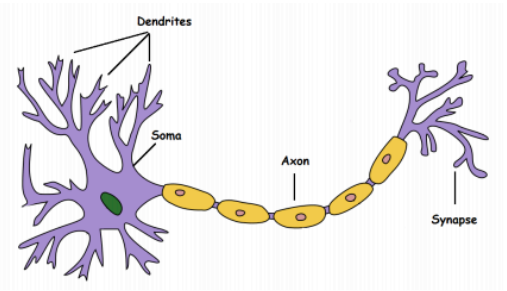
Deep Reinforcement Learning

SOTA Deep RL

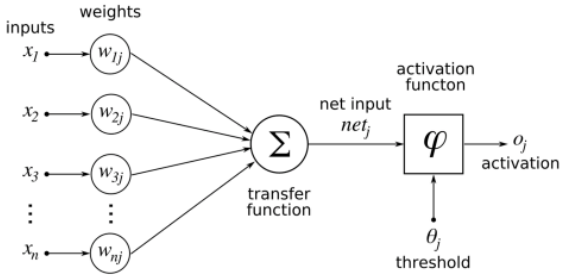
- Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor
- IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures
- Temporal Difference Models: Model-Free Deep RL for Model-Based Control
- Addressing Function Approximation Error in Actor-Critic Methods
- Learning by Playing – Solving Sparse Reward Tasks from Scratch
- Hierarchical Imitation and Reinforcement Learning
- Unsupervised Predictive Memory in a Goal-Directed Agent
- Data-Efficient Hierarchical Reinforcement Learning
- Visual Reinforcement Learning with Imagined Goals
- Horizon: Facebook's Open Source Applied Reinforcement Learning Platform

Neuromorphic Architectures

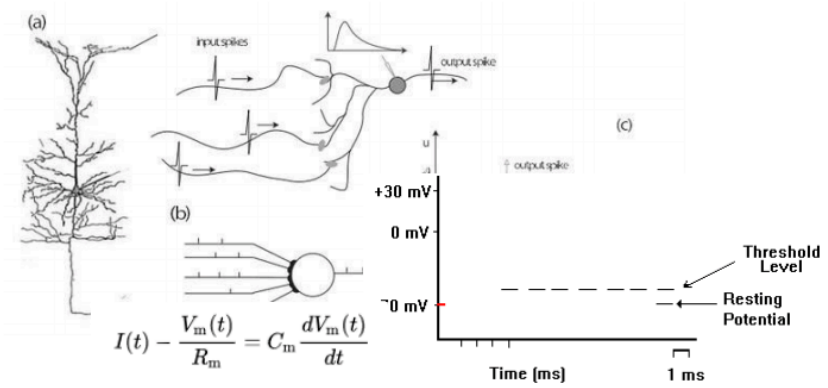
A Spiking Neuron is More Like a Biological Neuron



Deep Learning Neuron



Spiking Neuron



$$I(t) - \frac{V_m(t)}{R_m} = C_m \frac{dV_m(t)}{dt}$$

Neuromorphic Architectures

Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects

Rajesh P. N. Rao[1] and Dana H. Ballard[2]