11-695: AI Engineering
Unsupervision

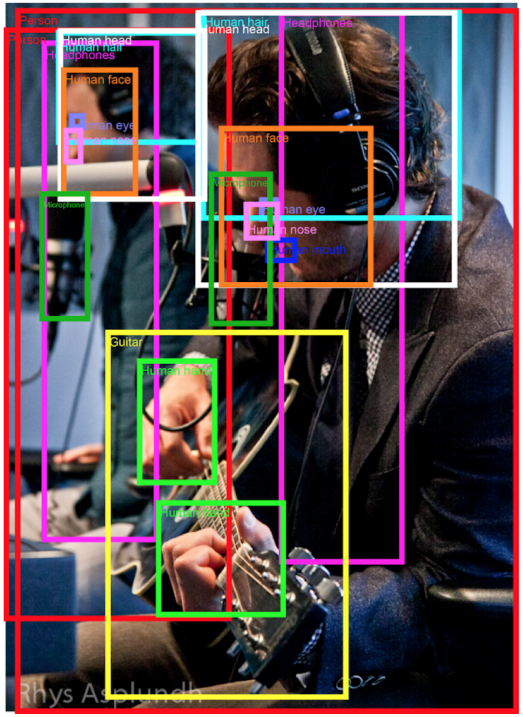
Spring 2019

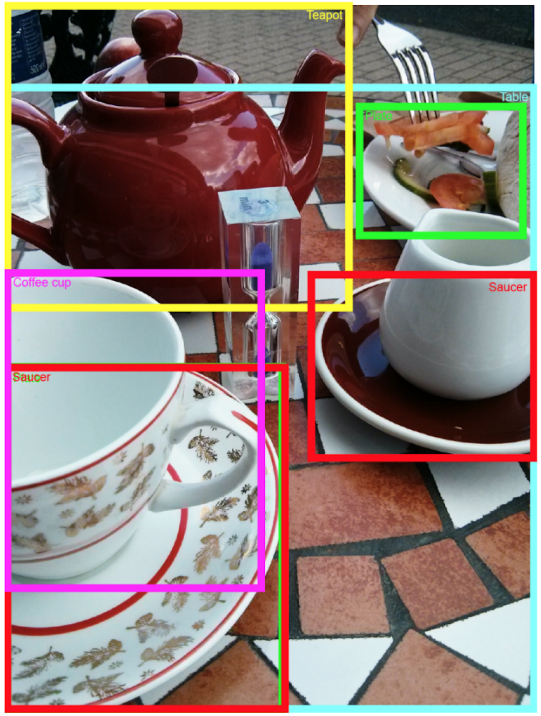
Outline

1. Data Issue
2. Methodology
3. Auto Encoder
4. Data Encoding to Embedded Representation
5. Decoding from Embedded Space

Problem with Labelled Data

Carnegie Mellon

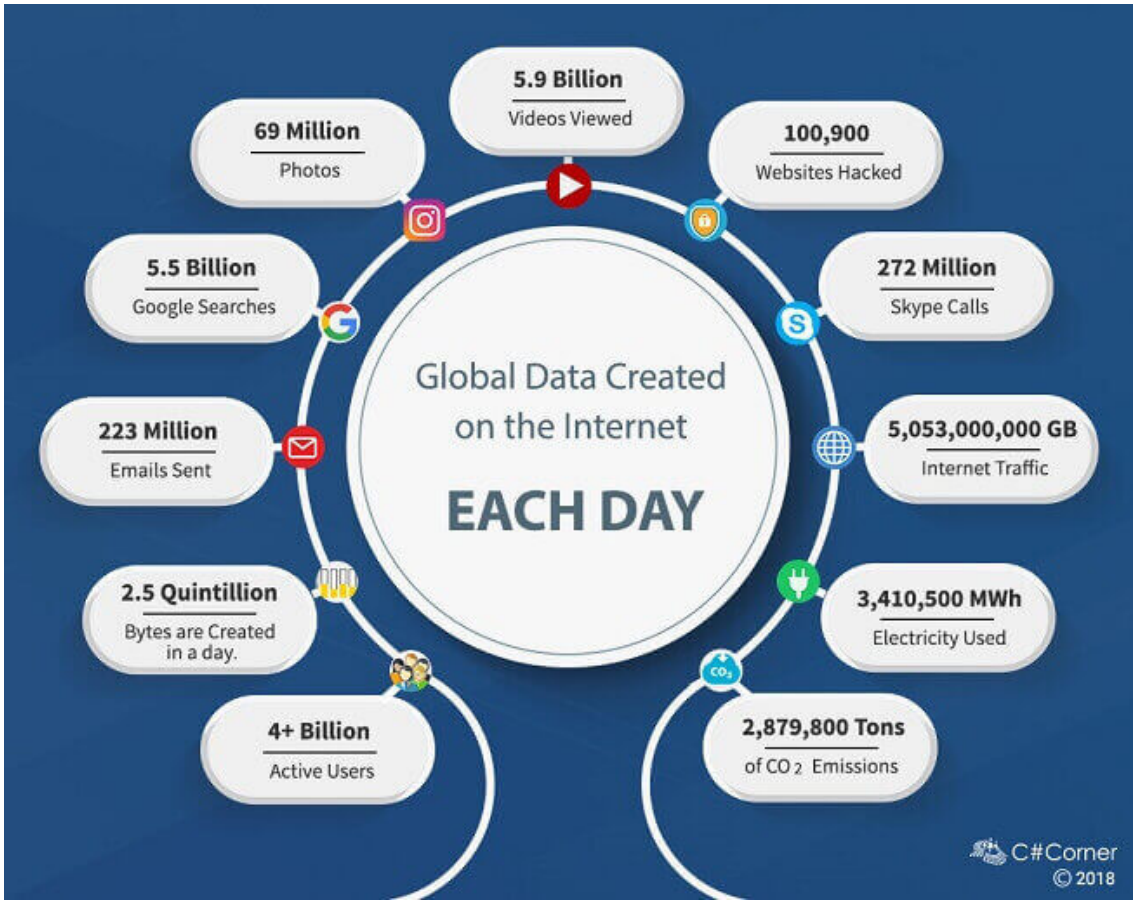




- Data is often carefully chosen,
- carefully cleaned and labelled,
- and usually *balanced*

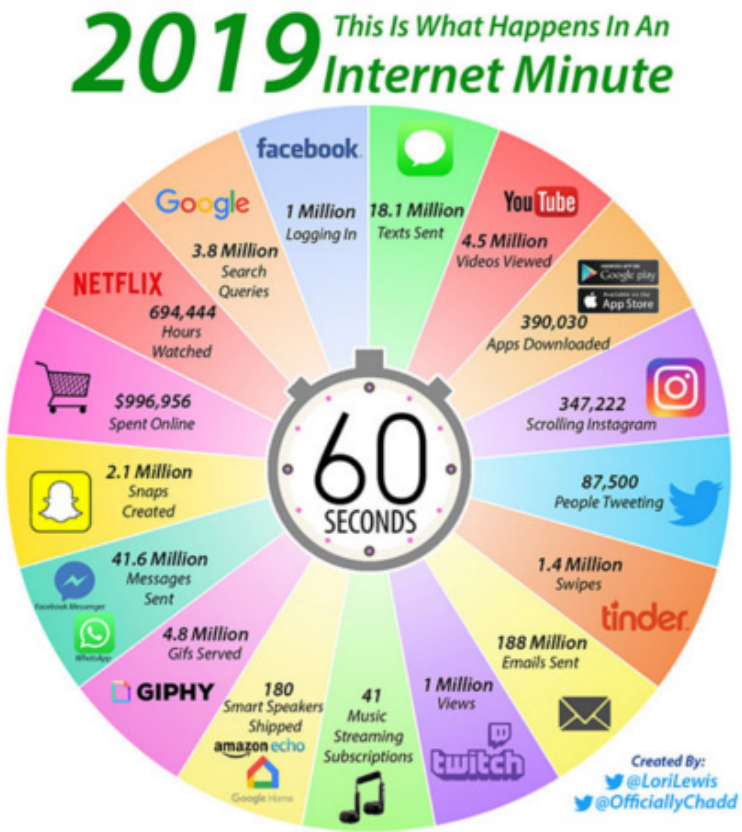
Image credit: Google AI Blog

But, The World's Data is Enormous



- Data is fast increased every second

Most Data is Unlabelled



- Also unstructured and unclean
- The cost to label them is prohibitively expensive.

Outline

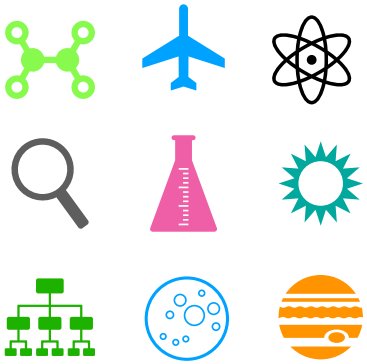
1. Data Issue
2. Methodology
3. Auto Encoder
4. Data Encoding to Embedded Representation
5. Decoding from Embedded Space

AI's Approach towards Big (Raw) Data

Carnegie Mellon

Raw Data

Model-based Features

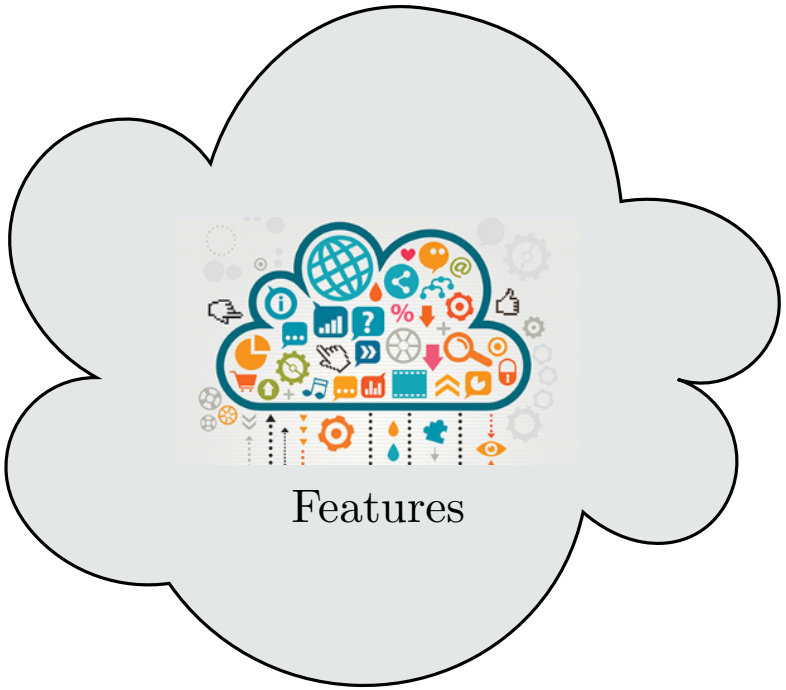


Tasks

- How about we not only focus on the task, but also put a proper focus on Data itself
- Difference from:
  - Fully labelled data: Supervised
  - Partially labelled data: Semi-supervised
- We have no labels thus no supervision → Unsupervised.

All is reduced to Feature Extraction

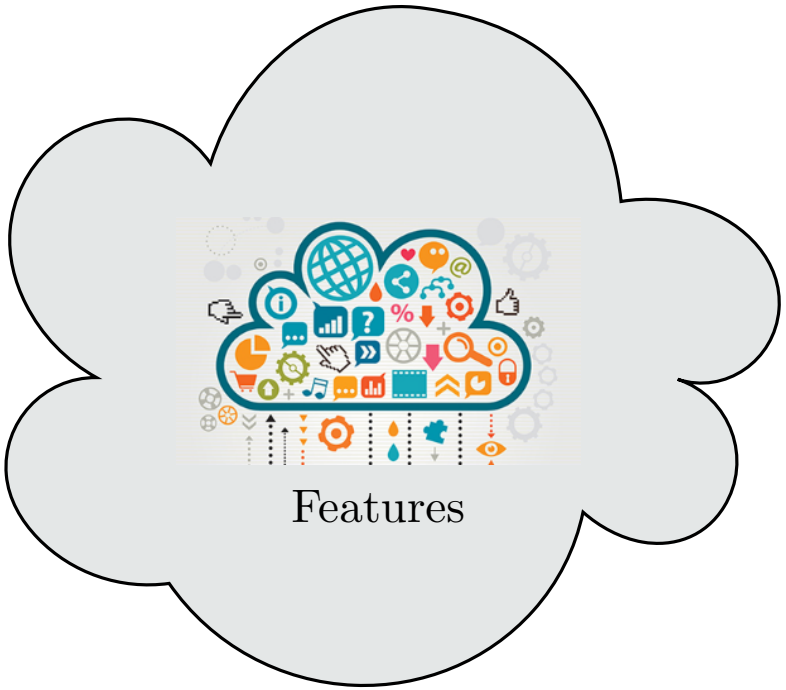
Raw Data



- Classification/Regression: use features to predict labels/numbers
- Required: ground-truth labels/numbers

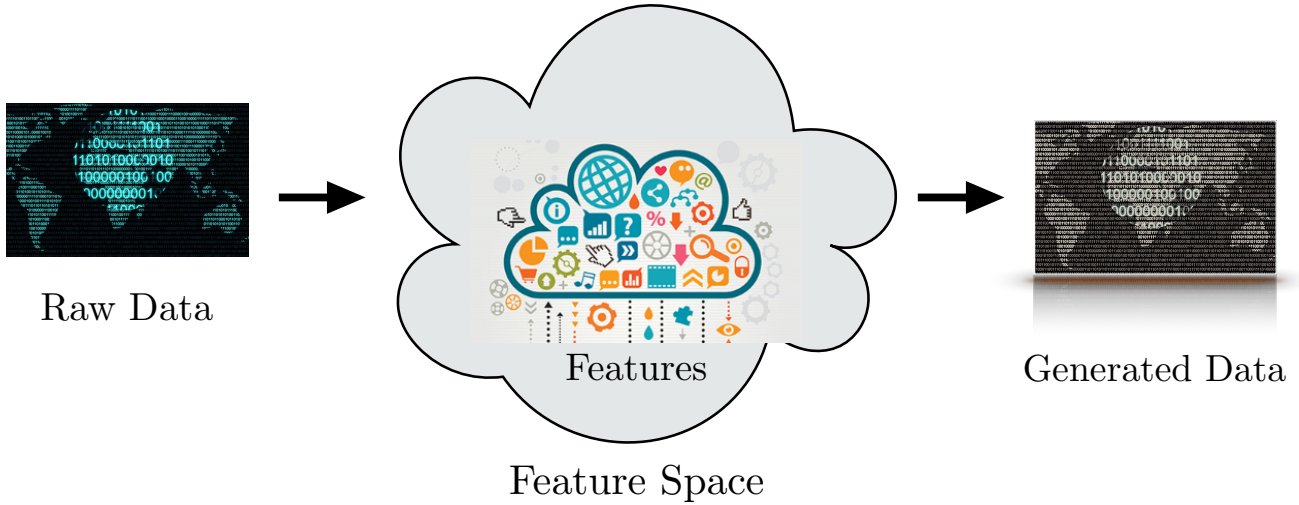
All is reduced to Feature Extraction

Raw Data



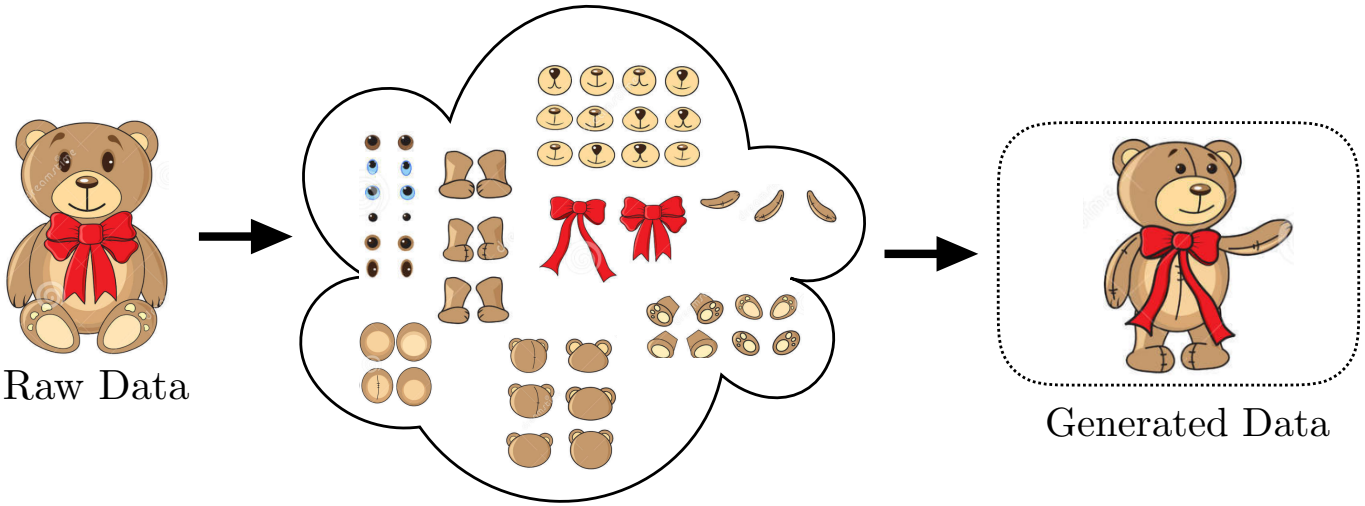
- Classification/Regression: use features to predict labels/numbers
- Required: ground-truth labels/numbers
- Unfortunately, we don't have such ground truths

Not really: Data are labels themselves



- Use each data sample itself as a label
  - Extract Features from x: $z = f_1(x)$
  - Reconstruct data $\hat{x}$: $\hat{x} = f_2(z)$
  - Minimize the distance $d(x, \hat{x})$, *a.k.a* reconstruction error.
- But, data itself are often highly-dimensional $\rightarrow$ (very) **hard!**
- Inject prior human knowledge/expertise about data, *e.g.*
  - Vision: use CNN as feature extractor
  - Text, sequential data: use RNN instead

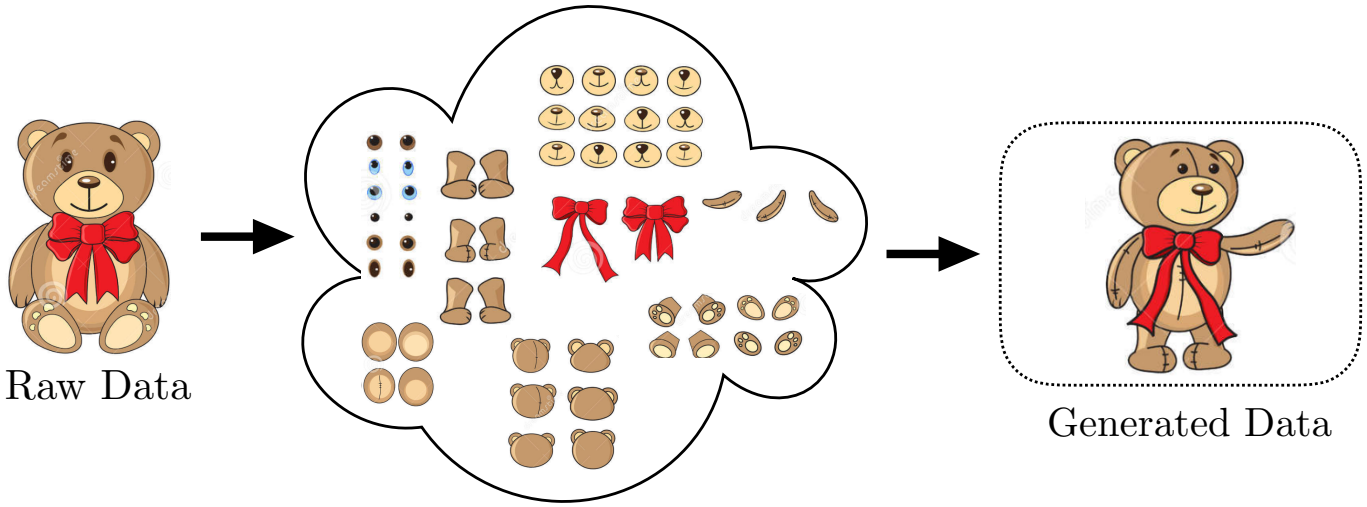
Generate a bear to just classify a bear?



- No
  - But we don't have labels
  - In many cases, we need to actually generate data
  - Or need to have good data features for our tasks
- Remind: for supervised, we don't need to generate data
  - Features needn't be well-structured to predict labels

Icons credit: Dreamstime.com

Generate a bear to just classify a bear?



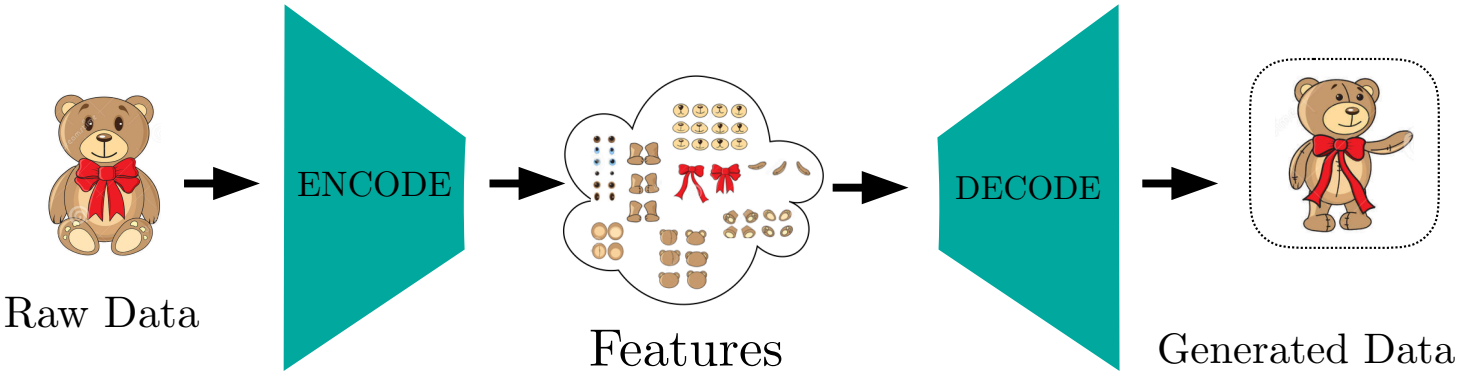
- Remind: for supervised, we don't need to generate data
  - Features needn't be well-structured to predict labels
- Comparison to Unsupervised in the classification/regression:
  - Supervised: we model $p(y|x)$
  - Unsupervised: $p(x, y)$

Icons credit: Dreamstime.com

Outline

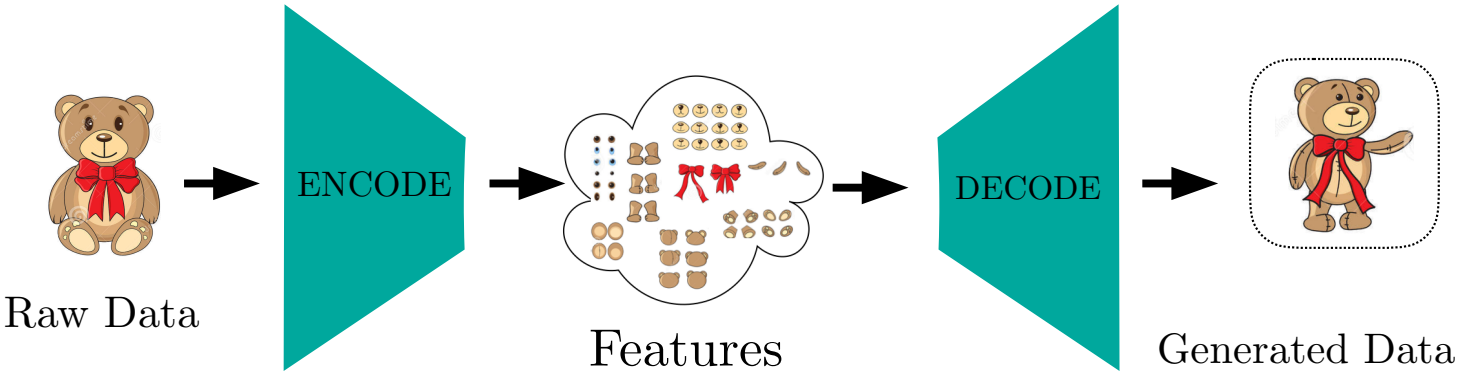
1. Data Issue
2. Methodology
3. Auto Encoder
4. Data Encoding to Embedded Representation
5. Decoding from Embedded Space

AutoEncoder



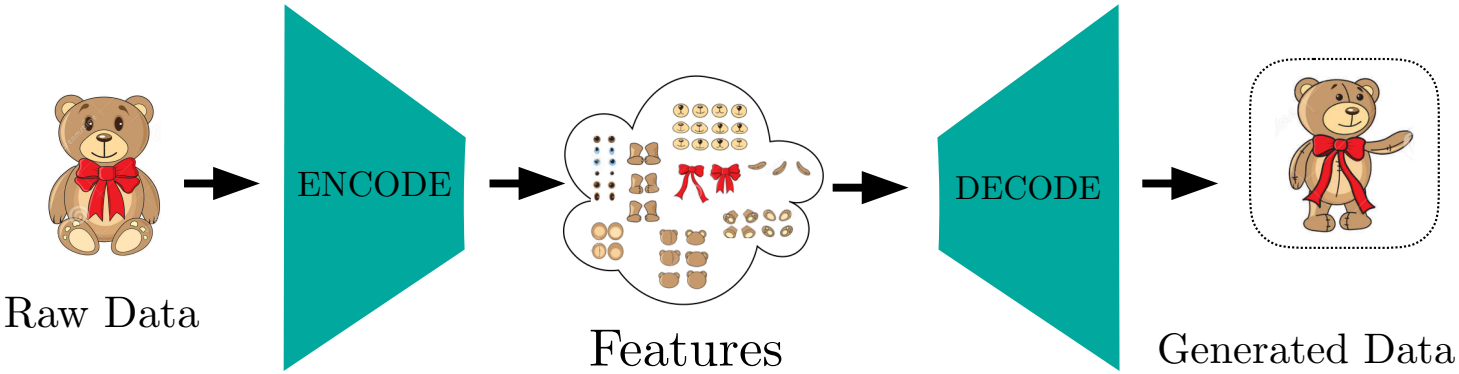
- Extract and Reconstruct *itself* → *Auto*-Encoder + Decoder
- What should the dimensionality of Features be?

AutoEncoder



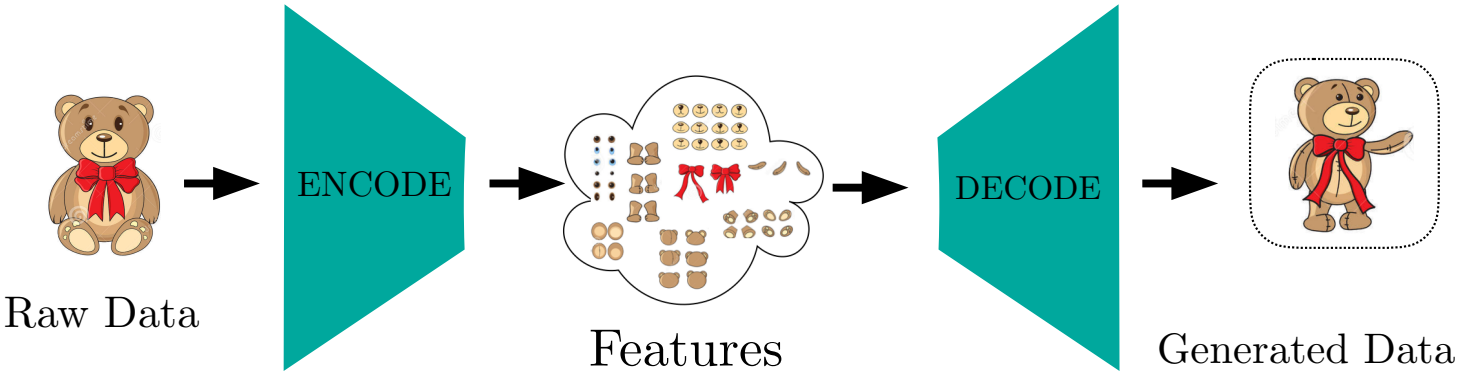
- Extract and Reconstruct *itself* → *Auto*-Encoder + Decoder
- What should the dimensionality of Features be?
  - Smaller than Input! Why?
  - So encoding (feature extraction) process is a compression

AutoEncoder



- Extract and Reconstruct *itself* → *Auto*-Encoder + Decoder
- What should the dimensionality of Features be?
  - Smaller than Input! Why?
  - So encoding (feature extraction) process is a compression
- Compression: we need to learn prominent features only
  - Yet should be enough to reconstruct data

AutoEncoder

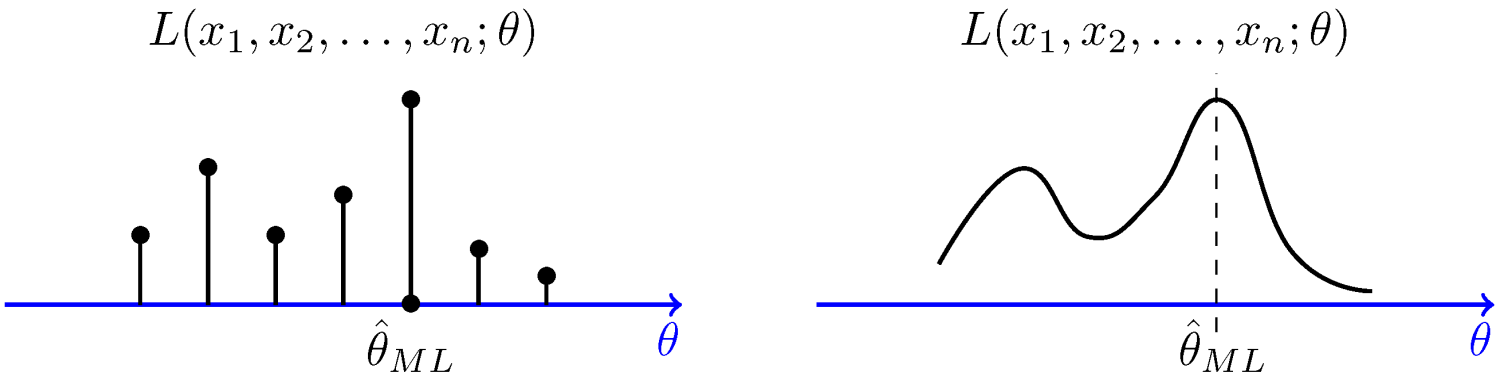


- Extract and Reconstruct *itself* $\rightarrow$ *Auto*-Encoder + Decoder
- What should the dimensionality of Features be?
  - Smaller than Input! Why?
  - So encoding (feature extraction) process is a compression
- Compression: we need to learn prominent features only
  - Yet should be enough to reconstruct data
  - How? Unless for trivial cases, we need params θ

Reduced to Feature Extraction (again)

How to *best* extract features from data?

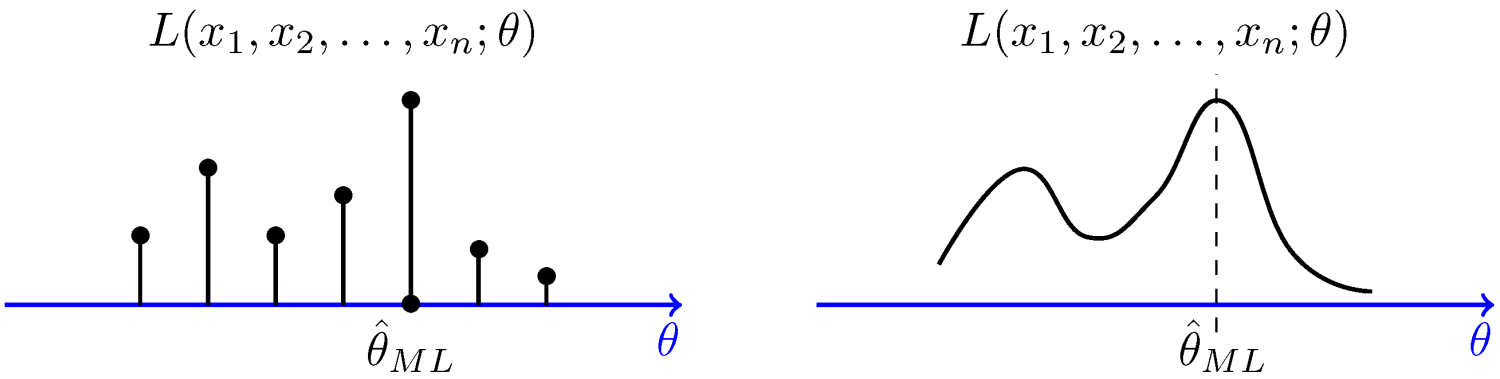
Relation: MLE Method



- input x, we need to find an estimator $\hat{\theta}$ best describing data
- Encode/Compress/Extract: $z = f_\theta(x)$
- Relation: Maximum Likelihood Estimation (MLE)
  - Likelihood: $\mathcal{L}(\theta; x) = p(x|\theta)$
  - Our job is to find: $\hat{\theta} = \text{argmax}_\theta \, \mathcal{L}(\theta; x)$

Image credit: probabilitycourse.com

Relation: MLE Method



- Trivial case, *e.g.* estimate a quadratic function (convex)
  - Take the first derivative
  - Set to zero and find the estimator from the solution
- Most cases: we need to approximate. How?

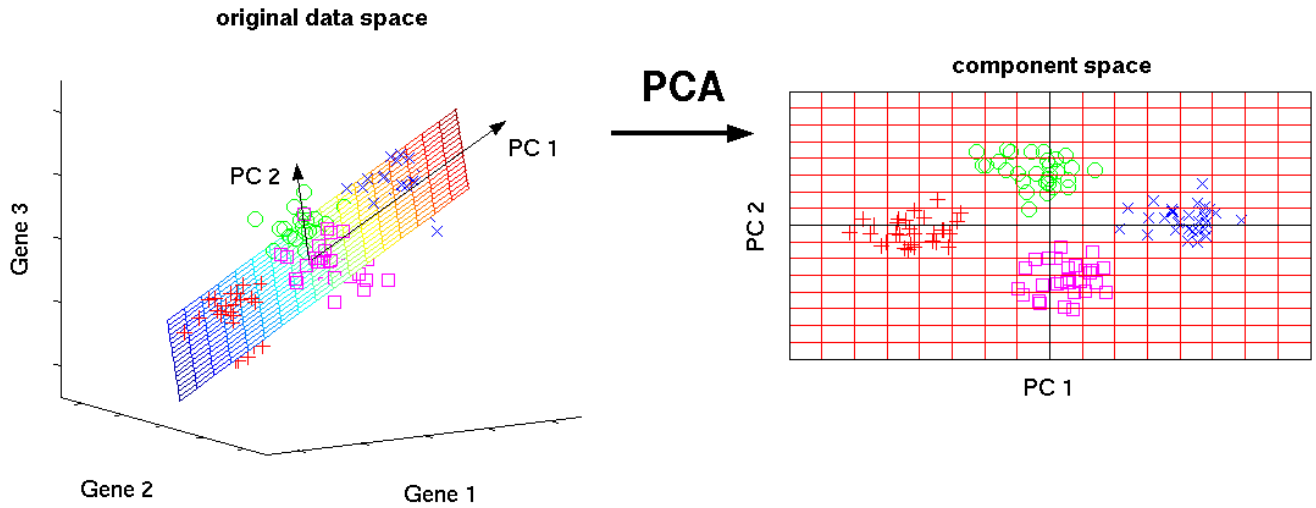
Image credit: probabilitycourse.com

Outline

1. Data Issue
2. Methodology
3. Auto Encoder
4. Data Encoding to Embedded Representation
5. Decoding from Embedded Space

Principal Components Analysis (PCA)

Carnegie Mellon



- Each Principal Component is a feature, and orthogonal to the rest
- Find a $s.t.$ $x^*a = \lambda a$ where λ is a scalar
  - One solution: perform SVD for x
  - Take only first few eigenvectors (hence approximation)

Image credit: Matthias Scholz, Ph.D. thesis

Principal Components Analysis (PCA)

Carnegie Mellon

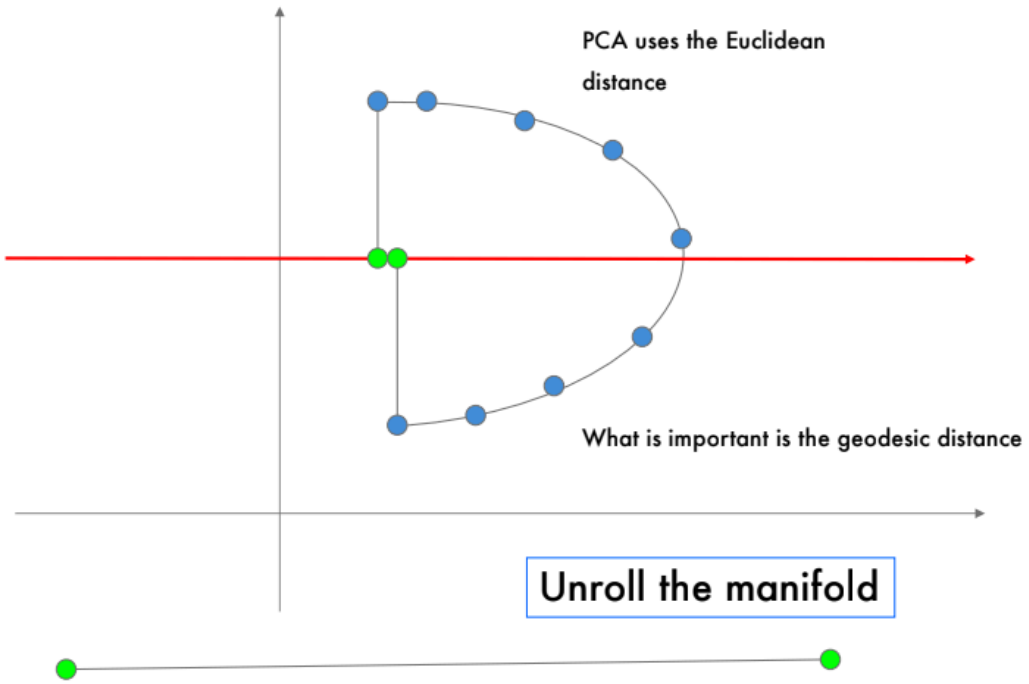
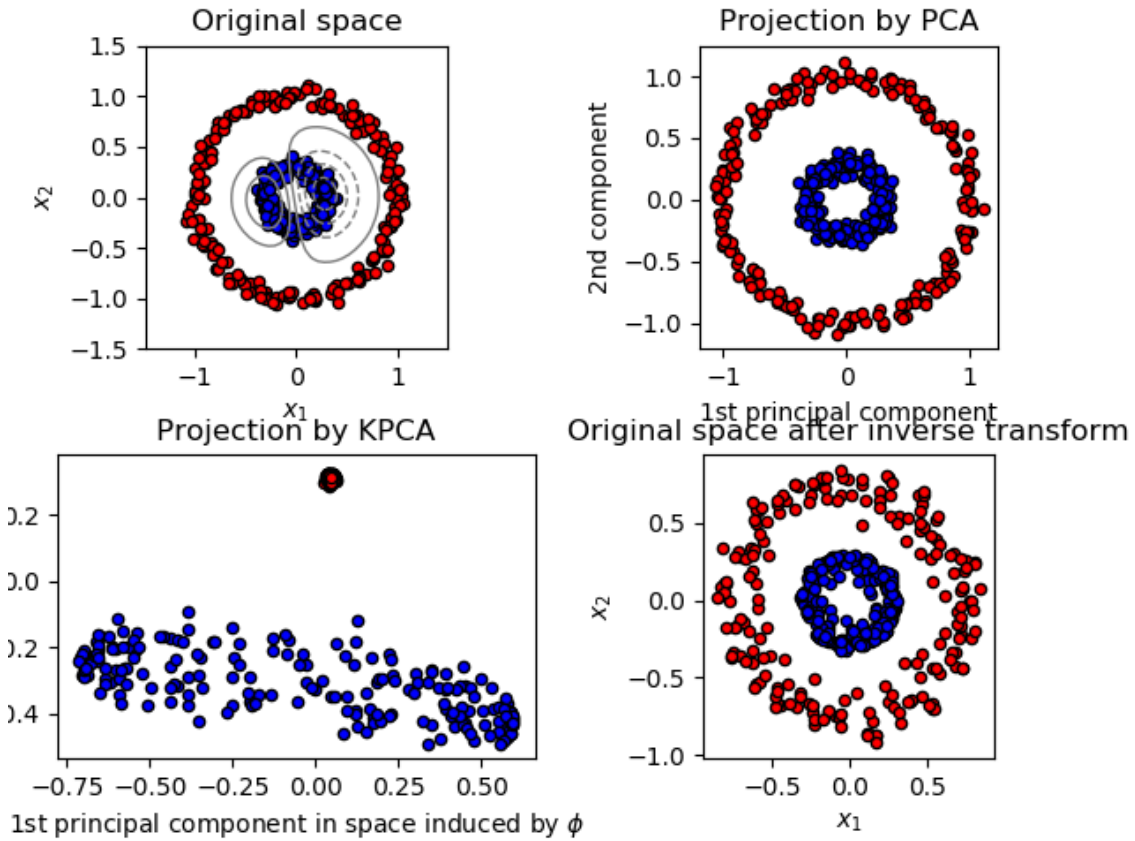


Image credit: Barnabas Poczos

Kernel PCA



- Recall: why do we use a kernel?
- Still a PCA, and have to carefully choose the kernel

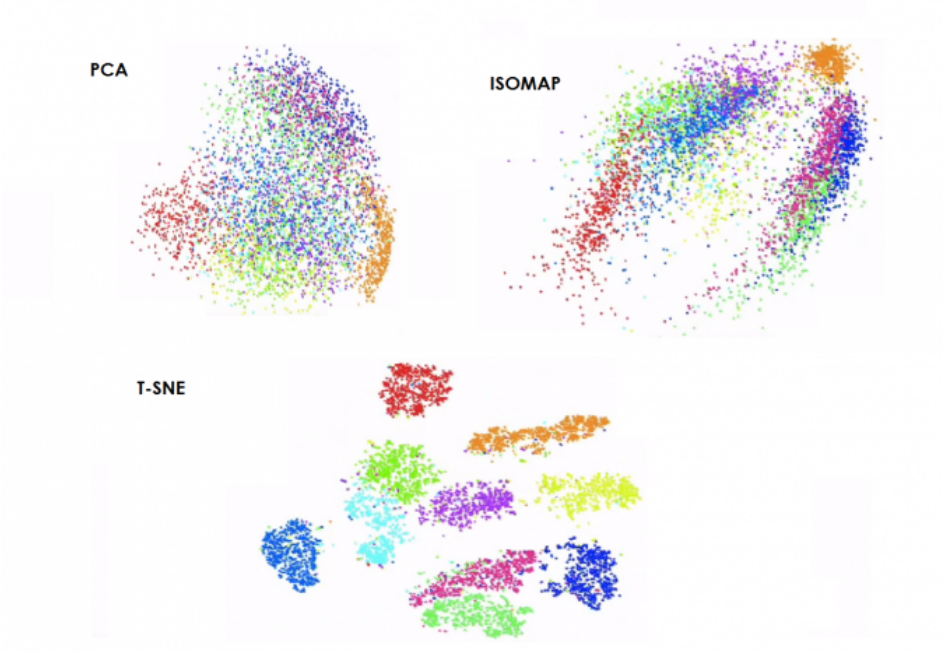
Image credit: scikitlearn

Decoded PCA (Generated Data)

- Number of eigenvectors taken: 2000, 1000, 500, 100, 50, 10, 2, 1

Image credit: Gene Kogan

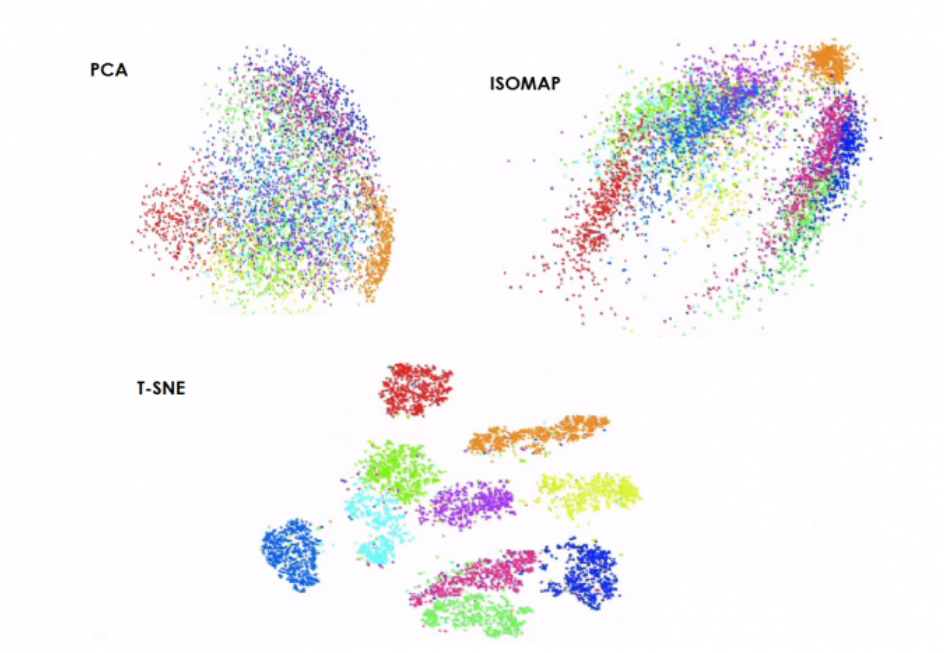
t-SNE



- Take into account relative distances between data points
- Is a non-linear method
- Both PCA and t-SNE suffer in high dimensional space.

Image credit: quantdare.com

t-SNE



- Take into account relative distances between data points
- Is a non-linear method
- Both PCA and t-SNE suffer in high dimensional space.
- *It's time for Deep NN to be used!*

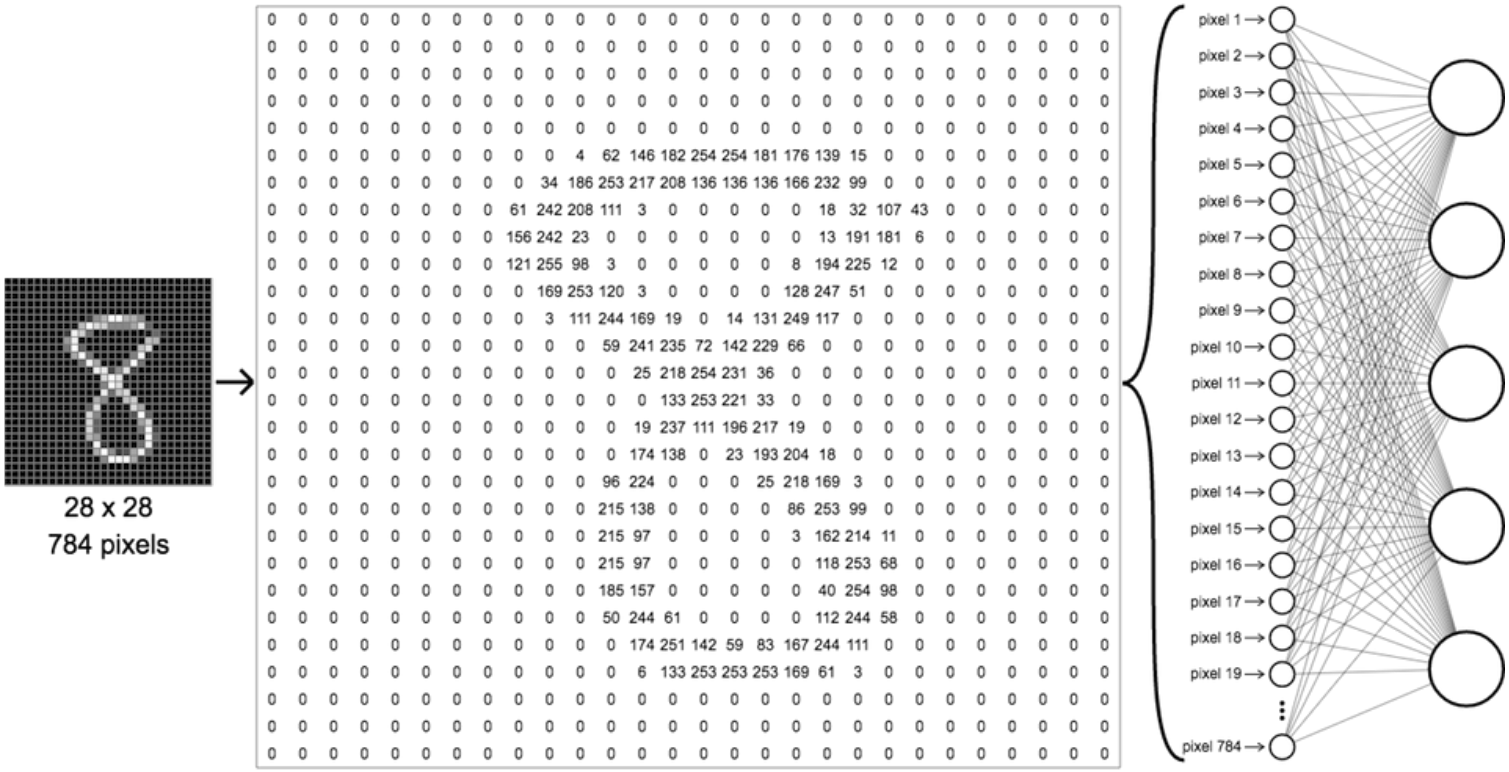
Image credit: quantdare.com

t-SNE

- Original space:
  - one-sided distance: $p_{i|j} = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$
  - two-sided (gaussian centered density): $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$
- Embedded space: similar distance: $q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq i}(1 + \|y_i - y_k\|^2)^{-1}}$
- Target: by mimimizing the KL distance:

  $y_1, ..., y_D = \text{argmin}\, KL(P\|Q) = \text{argmin} \sum_{i \neq j} p_{ij}\, log \frac{p_{ij}}{q_{ij}}$

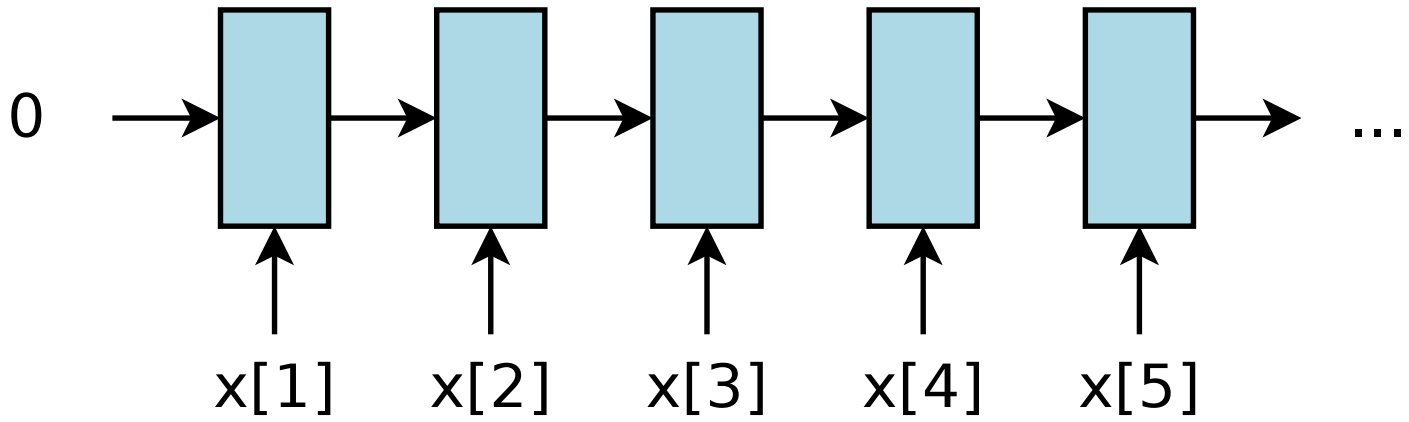
General Case: Use MLP



- Caveat: need to flatten the input
- With data knowledge, we can do a better job

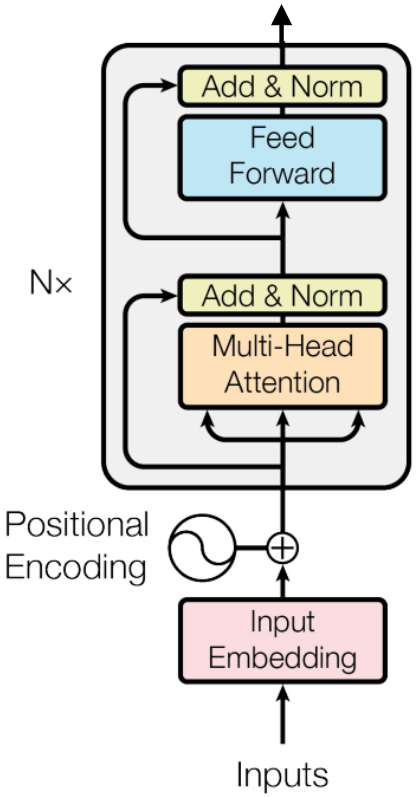
Image credit: ml4a.github.io

Sequential: Use RNN



- Representation has sequential correlations

Sequential: *Transformer*

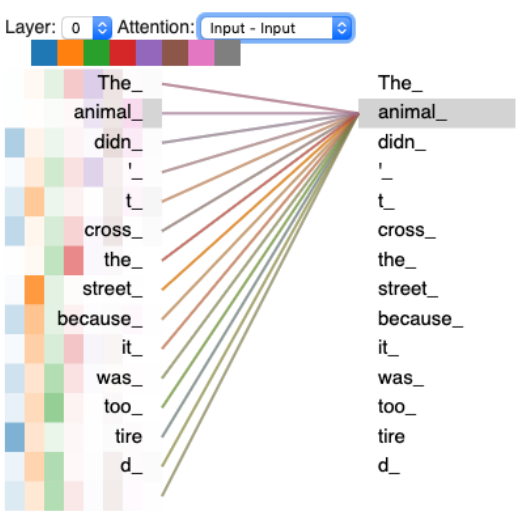


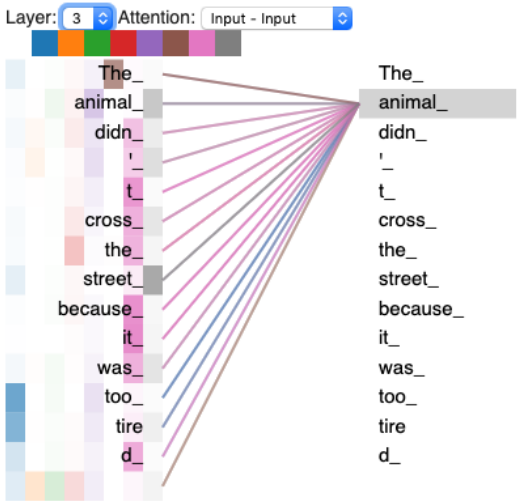
- We don't rely on RNN function anymore
- Instead, rely on $N*$ (multi-head attentions + MLP) blocks
- And more ...

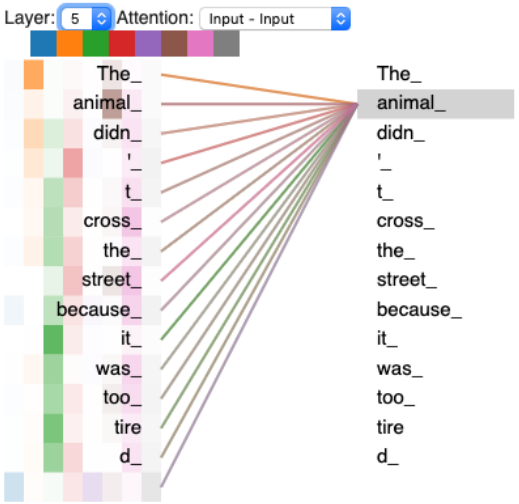
Image credit: Ashish Vaswani *et al.*

Self-Attention (of *it* to *itself*)

Carnegie Mellon





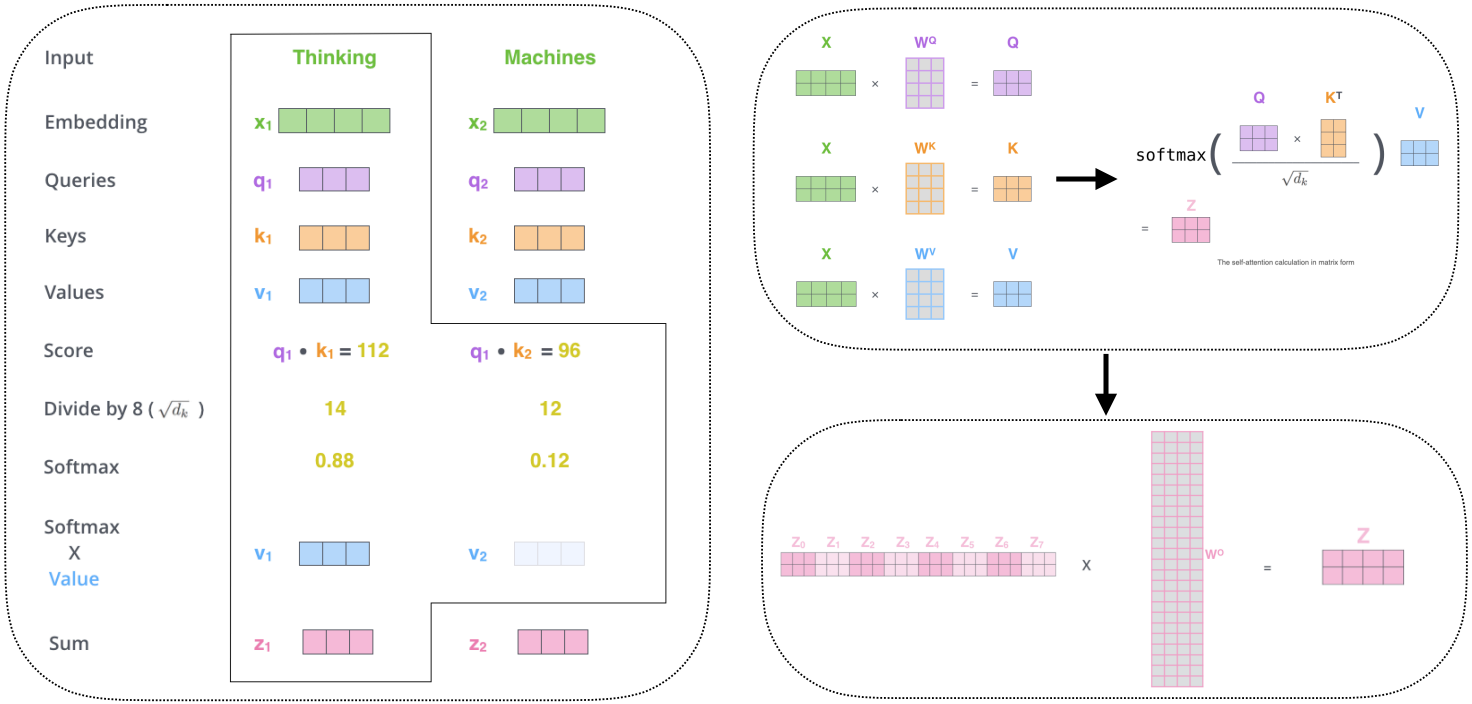


- Instead of encoding the sequence into a single hidden vector
- Now we calculate the relationship of each word vs. *every* another word
- Maybe a better idea for languages in which the order is very flexible

Image credit: Tensor2Tensor Ipython Notebook - Google

Multi-head Attention

Carnegie Mellon

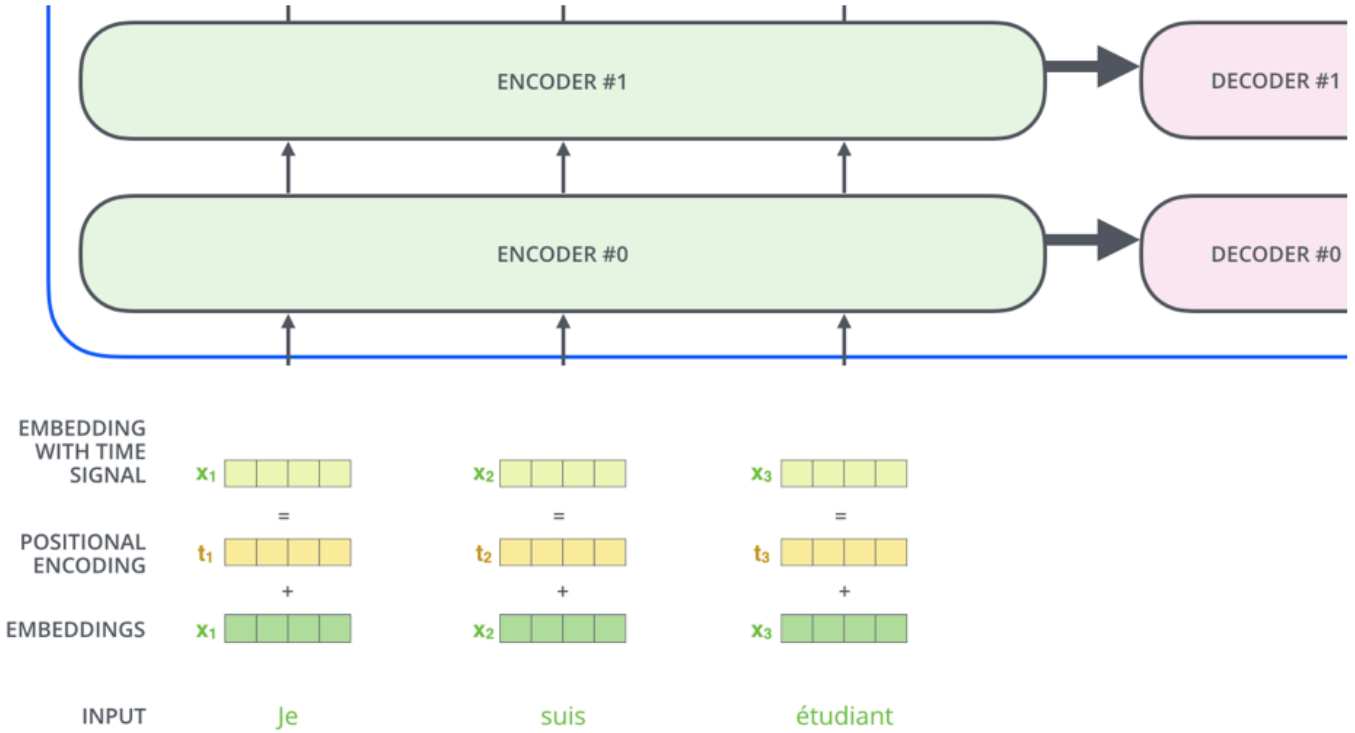


- Instead of using a single FC, use a triple (q, k, v) (query, key, value)
- Normalized with input length and with softmax
- Merge all head using concatenation and a FC with weight W^0

Image credit: Jay Alammar

Multi-head Attn Able to Replace RNN?

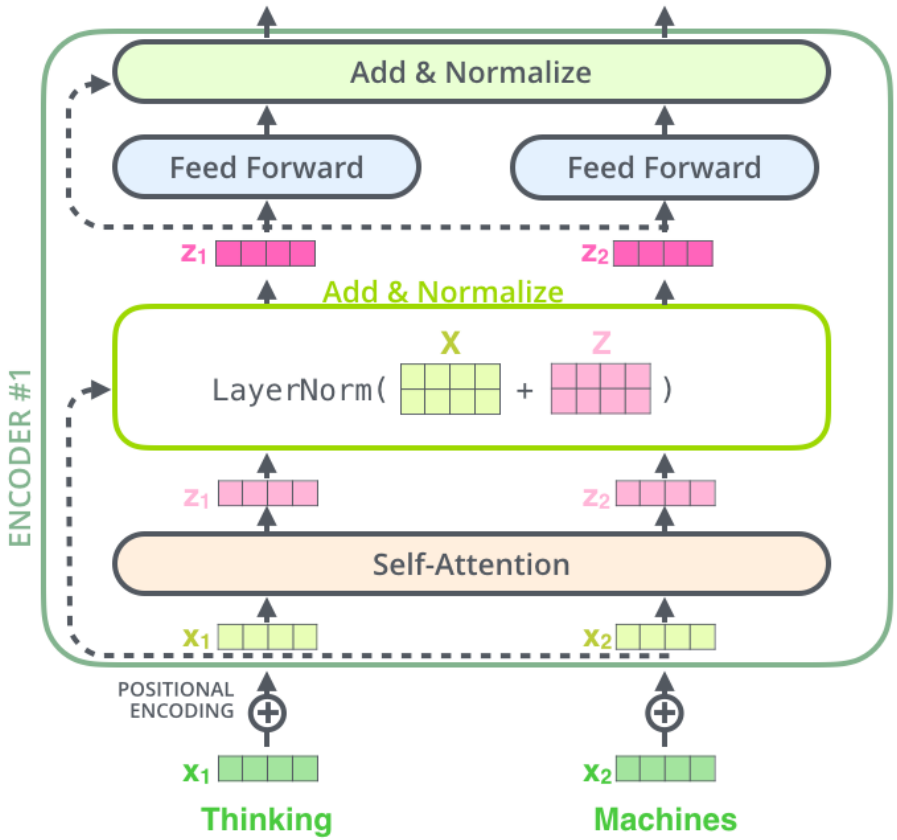
Carnegie Mellon



- Not yet, needs an alternative way to model sequence
- *Positional Encoding* is added to word embedding

Image credit: Jay Alammar

Putting together: Transformer Encoder

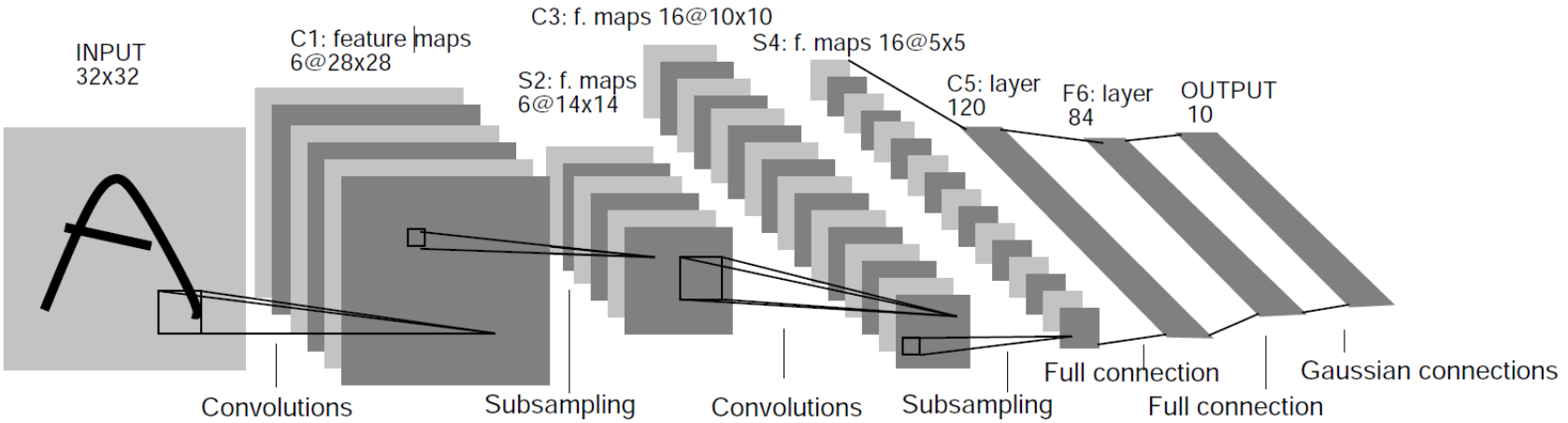


- Residual connection is used at self-attention and MLP
- Batch Norm is the last layer.
- Note: it also stacks multiple encoders (and decoders).

Image credit: Jay Alammar

Image/Video: Use CNN

Carnegie Mellon



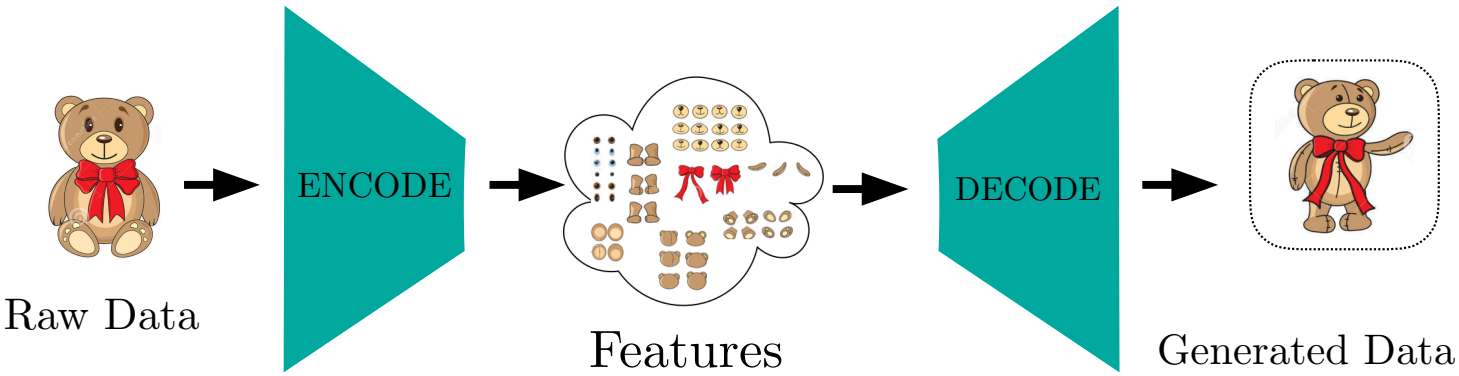
- Encoded representation has spatial correlations

Image credit:

Outline

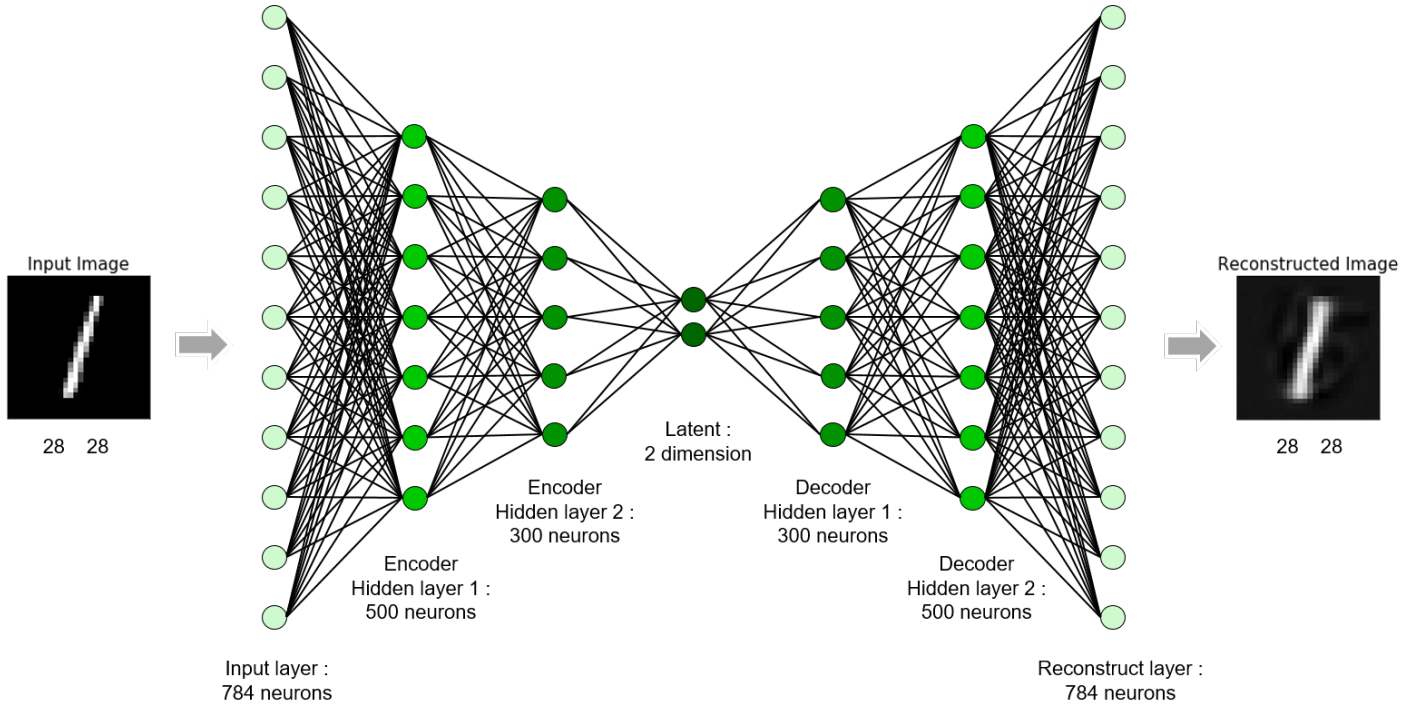
1. Data Issue
2. Methodology
3. Auto Encoder
4. Data Encoding to Embedded Representation
5. Decoding from Embedded Space

General case



- Perform the inverse process of decoder
  - Encoder: compress from dimension N to D for N > D
  - Decoder: uncompress from D back to N

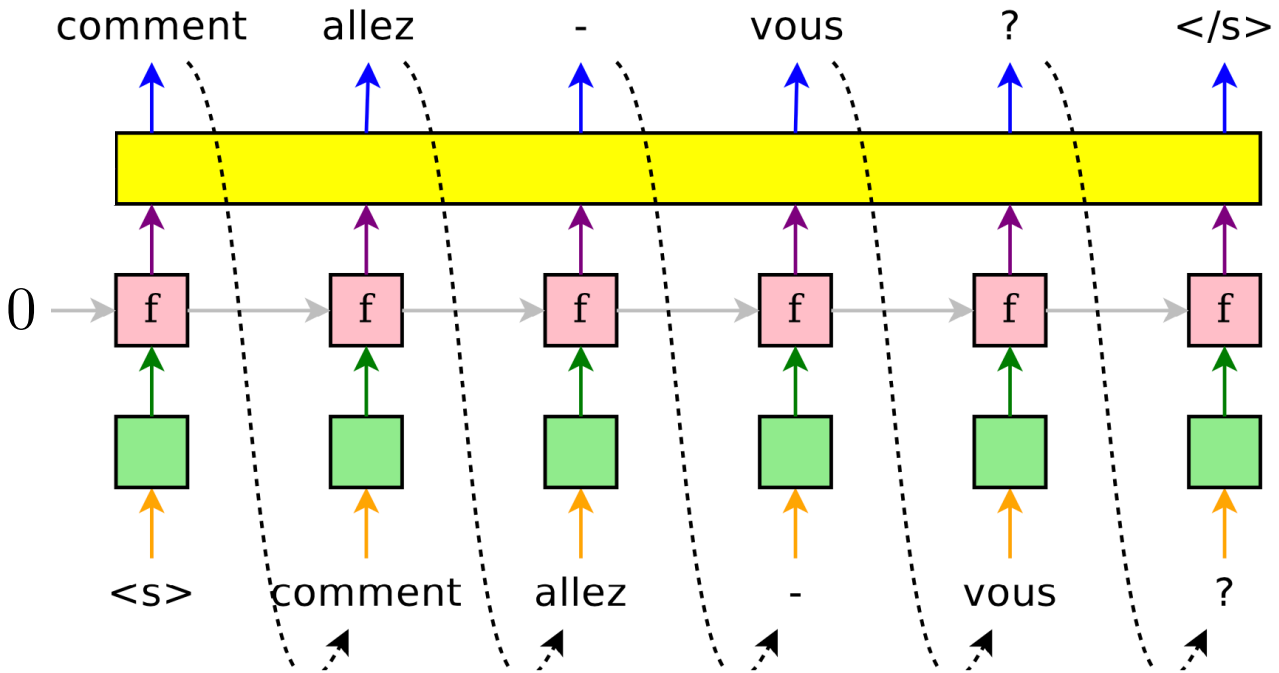
MLP Decoding



- Perform the inverse process of decoder
  - Encoder: MLP with decreasing dimensions
  - Decoder: MLP with increasing dimensions to the original
- Tutorial: ▸ MNIST AutoEncoder Tensorflow

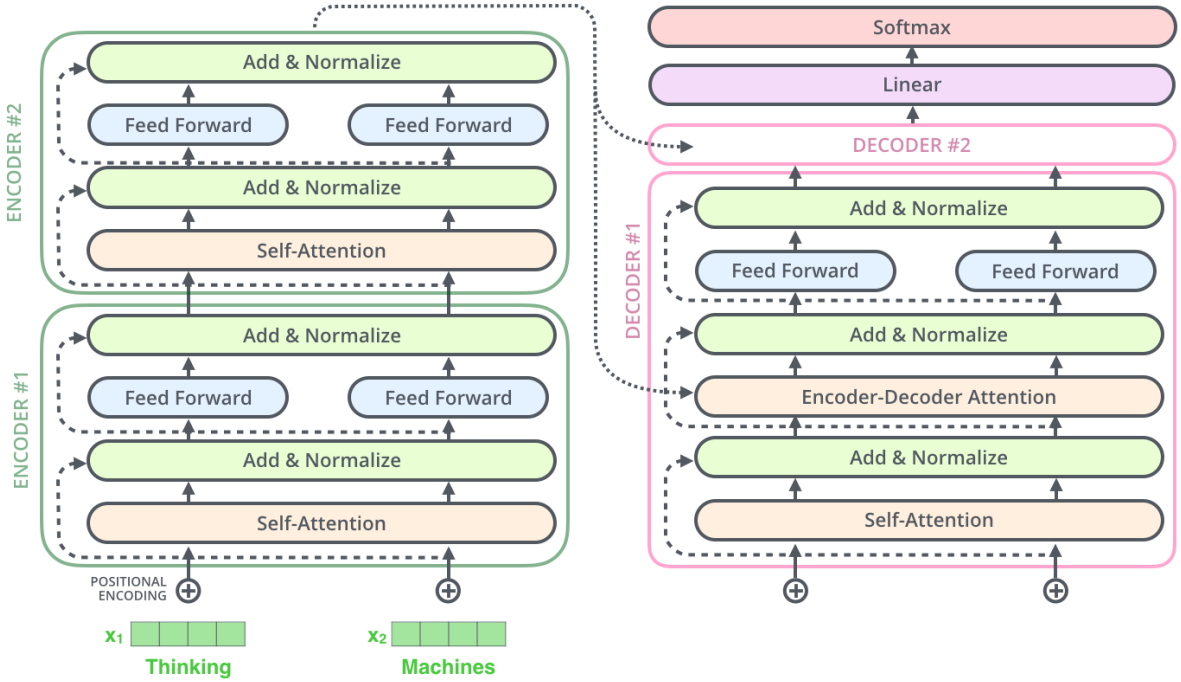
Image credit: i-systems.github.io

RNN Decoding



- Perform the inverse process of Encoder
  - Using previous output and previous hidden
  - It models $p(x_t|x_{<t}$ so sometimes it's called an *auto-regressive model*.
- Tutorial: ▸ RNN Text Generation with Eager Execution

Transformer Decoding

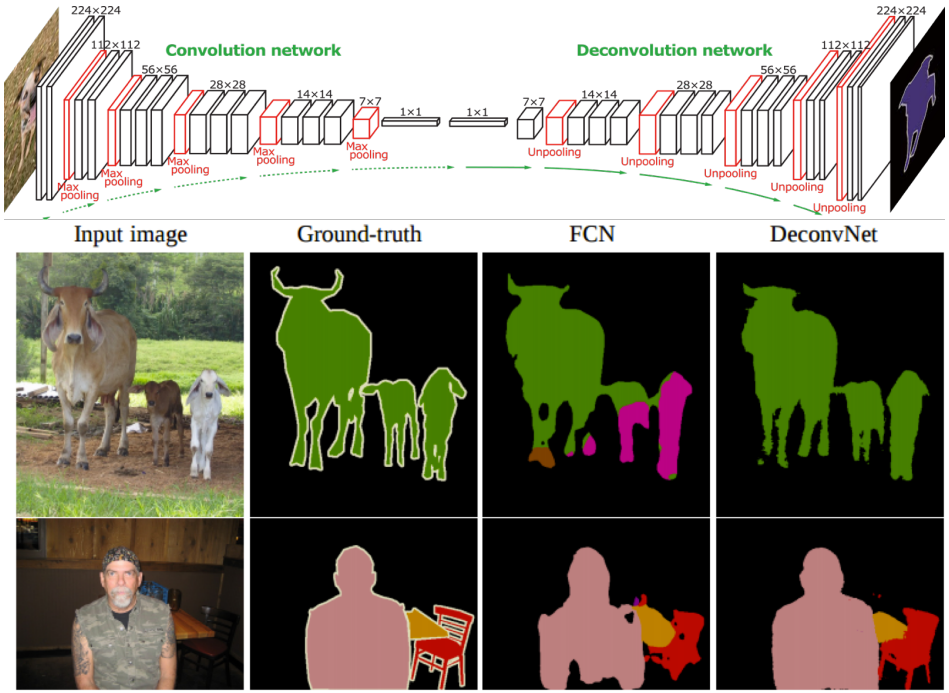


- Self-Attention is used, but only for the seen sequence to that point
- Then normal Attention is used as in Seq2Seq
- Decoding as usual to find a word in vocabulary.
- Tutorial: ▸ NMT-Transformer with pre-trained weights

Image credit: Jay Alammar

CNN Decoding: Beyond MNIST

Carnegie Mellon

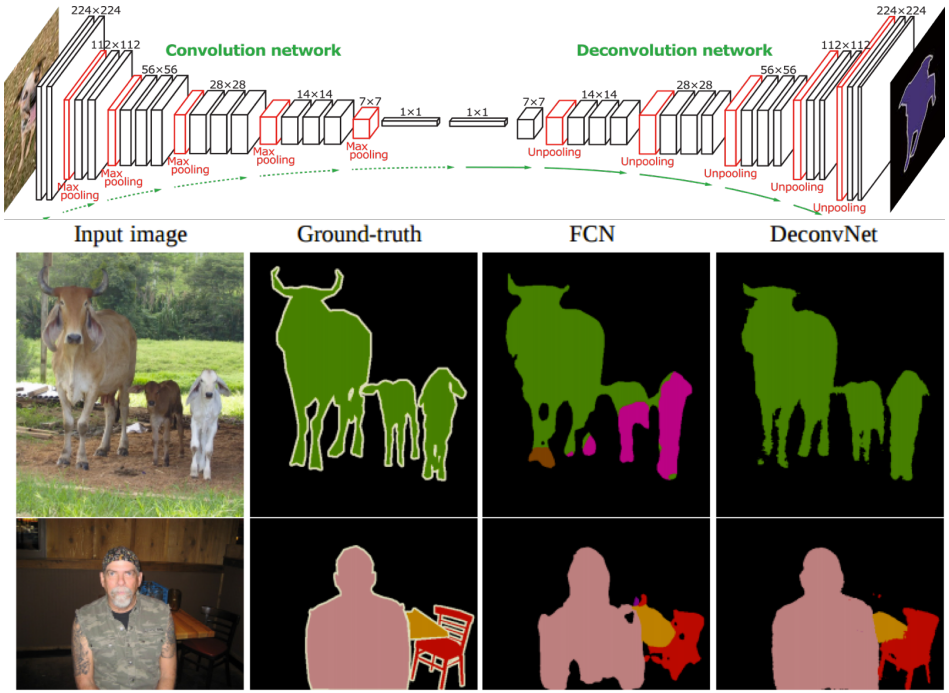


- Deconvolution is only good enough for such tasks as segmetation where detail is not the first priority
- Maybe we need to add some constraint into embedded space?
- Or maybe we should not use MLE-based method?

Image credit: Hyeonwoo Noh et al.

CNN Decoding: Beyond MNIST

Carnegie Mellon



- Maybe we need to add some constraint into embedded space?
- Or maybe we should not use MLE-based method?
- *Next lectures will help provide some appealing solutions to those questions.*

Image credit: Hyeonwoo Noh et al.