ADL QA

04082023

Tim Mondorf, DIKU, CJK681

KØBENHAVNS UNIVERSITET

1. Recap

Predict XOR with a Multi-Layer Perceptron (MLP)

- Explain the basics of the model
- Explain the concepts Soft-max activation function, mini-batch learning, vanishing gradient, RMSprop update rule, "single layer with enough neurons"

2. CNNs, U-Nets, segmentation

CNNs

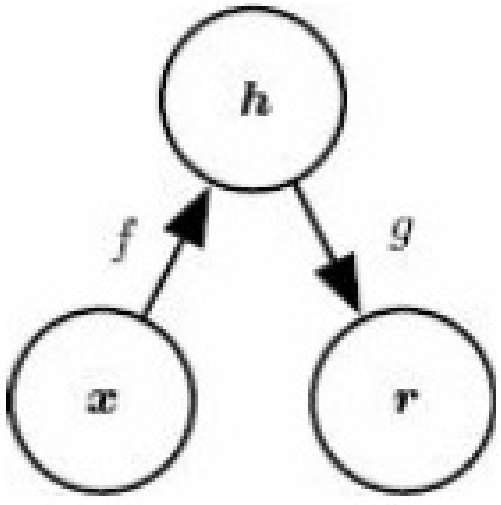
- What are CNNs and how are they different from normal neural networks, how are they computationally more efficient, how is transposed convolution applied to manage colors, what is the relationship to the Haar Cascade from 2001, what is spatial invariance, translation invariance or translational equivariance
- What are UNets and how are they an improvement over CNNs
- What is mini-batch normalization, why is it used, how is it different from training mode to prediction mode

3. Generative models, VAEs, GANs. Goodfellow chapter 14 Autoencoders

Goodfellow chapter 14

Goodfellow chapter 14 autoencoders and 14.1 undercomplete autoencoders

- What is an autoencoder
- How is PCA is a special case of the autoencoder
- How do we prevent an autoencoder from learning the identity function
- The simplest case is to make the autoencoder shallow and make the code size small
- Minimize $L(x, g(f(x)))$



Goodfellow chapter 14.2 Regularized autoencoders

- What if we want a deep autoencoder with a lot of code, i.e. an overcomplete autoencoder?
- That is what a regularized autoencoder accomplishes?
- What are the three ways of preventing an overcomplete autoencoder from just learning the identity function - These are detailed in 14.2.1, 14.2.2 and 14.2.3
  - sparseness of the representation. This is something other than low dimensionality of model (14.2.1)
  - robustness to noise (14.2.2)
  - and smallness of the derivative (14.2.3)?

Goodfellow chapter 14.2.1 Regularized autoencoders

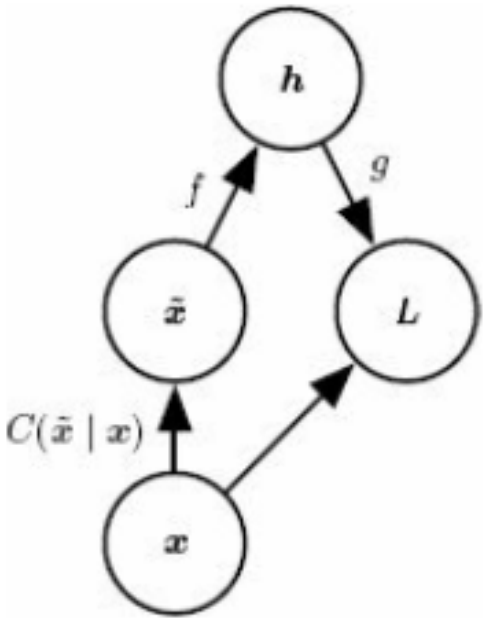
- A sparse autoencoder will minimize $L(x, g(f(x))) + \Omega(h)$
- What is the maximum likelihood view of the sparse autoencoder?
- ML means to select the parameter values of $\theta(x)$ so that we maximize $\theta(x)|x$, since x is now observed. For simplicity of notation, we now call them θ and x, so we are asked to maximize $p(\theta|x)$
- According to Bayes theorem $p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$
- Take the logarithm on both sides
  $logp(\theta|x) = logp(x|\theta) + log(\theta) - logp(x)$

Goodfellow chapter 14.2.1 Regularized autoencoders contd

- Since x is observed, $p(x) = 1 \Leftrightarrow logp(x) = 0$
- So maximum likelihood of the autoencoder becomes $maxlogp(x|\theta) + log(\theta)$
- The very naive interpretation is that probabilities are smaller than 1 so $log(\theta)$ is a negative term. A complex model with many small probabilities will optimize probability but introduce many highly negative terms in the optimizing function. For example, the sum of the logarithms of two probabilities of 0.5 is -1.4 whereas the sum of the logarithms of 10 probabilites of 0.1 is -23. This is not how Goodfellow explains it
- Rather, he says that the encoding function h can be defined as inducing sparsity
- so the modelling of the ith outcome is $p(h_i) = \frac{\lambda}{2} e^{-\lambda|h_i|}$

Goodfellow chapter 14.2.2 Robustness to noise - Denoising autoencoders

- Loss-function now becomes $L(g(f(\hat{x})), x)$



Goodfellow chapter 14.2.3 Smallness of the derivative - regularizing by penalizing derivatives

- Goodfellow makes this idea more explicit by defining the loss function as
- $L(x, g(f(x))) + \Omega(h, x)$ where $\Omega(h, x) = \lambda \Sigma_i ||\nabla_x h_i||^2$
- i presumably refers to the various layers of the model
- This is referred to as a contractive autoencoder

Goodfellow chapter 14.3. Representational power, layer size and depth

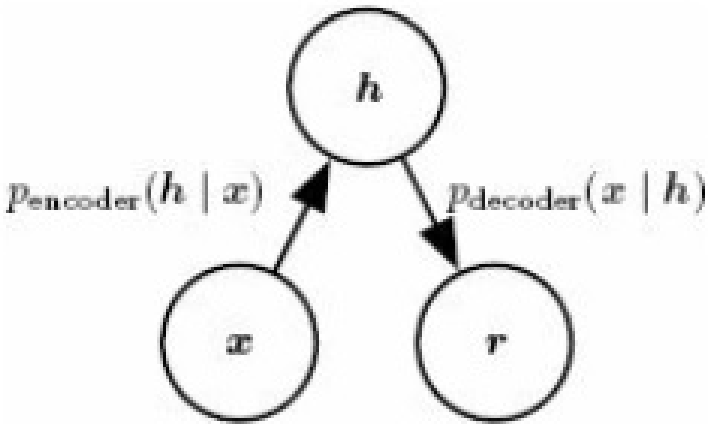
- Universal approximator theorem
  - Any function that maps from R to R or from R^N to R^N can be approximated by a neural network with only one layer if there is not restriction on the number of hidden ndes in this layer
  - Since neural networks are themselves functions, a neural network of any depth can be approximated by a network of only one layer (in the worst case, one node for each member of the training set)
  - The weights activation functions of these two networks need not be the same - so it is not the question of just reorganising the nodes
- ML has 'no free lunch' - no guarantee that any model performs better than others on all input sets. But empirically, deep models generalize well
- Deeper models perform well not just because they have more nodes, but also because they have skip-layers or because they do not connect all nodes in two given layers

Goodfellow chapter 14.3. Representational power, layer size and depth contd

- contd - so deep networks have so more structure and more degrees of freedom
- Deeper networks reflect a belief that we are trying to learn a function that is composed of several simpler functions
- Deep networks therefore approximate computer programs
- Deeper networks have computational benefits
- Now back to chapter 14 - what does all this mean for encoders and decoders
  - Encoder and decoder layers can be DEEP
  - Goodfellow then in chap 14 states without substantiation
  - Deep neural networks can exponentially reduce the computation cost of representing 'some functions'
  - Deep neural networks can exponentially reduce the required amount of training data (two latter statements not in chap 6)

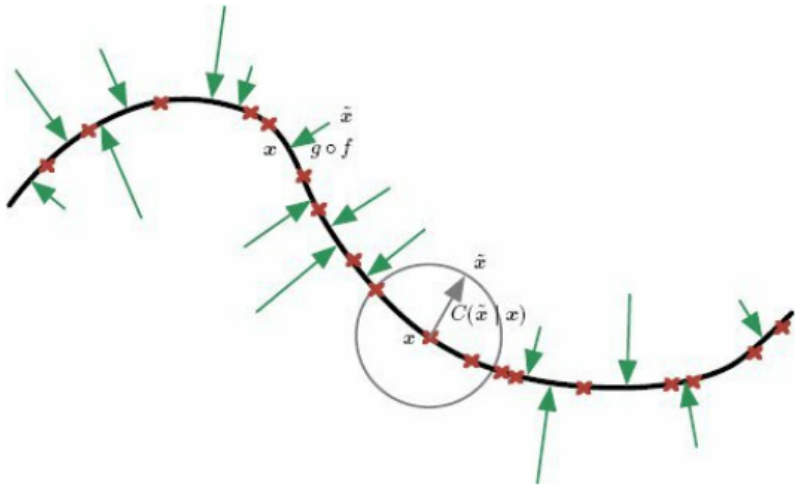
Goodfellow chapter 14.4 Stochastic encoders and decoders

- Already mentioned above
- Best understood in the later denoising section
- Now both encoder and decoder inject noise so the model now defines a stochastic encoder $p_{model}(h|x)$ and a stochastic decoder $p_{model}(x|h)$



Goodfellow chapter 14.5 Denoising autoencoders

- Difficult to see delineation between Goodfellow 14.4 and 14.5. Denoising autoencoders are the only logical application of stochastic encoders since stochastic encoders in their present form do not allow us to understand the function directly
- First graph shows the noise addition (Fig 14.4 Goodfellow)



Goodfellow chapter 14.5 Denoising autoencoders contd

- Similar to stochastic gradient descent - we are dealing with imperfect materializations of functions and imperfect techniques to estimate them
- Functions are non-linear and may be highly uneven - so gradient descent is already not a deterministic process
- Goodfellow briefly discusses an alternative estimation technique called score matching. Instead of minimizing loss function or maximizing likelihood, a good learned function is a function that is trained on the corrupted input but which exhibits the same gradients as the uncorrupted input
- Not clear why G has this detour into yet another estimation technique, but the gradients are interesting and can be interpreted

Goodfellow chapter 14.5 Denoising autoencoders contd

- G calls the observed data a manifold - so in two-dimensions, the data can be represented by a one-dimensional path in two-dimensions. There is no guarantee that this is the case, but if we have a significant relationship between the two variables, it will be the case
- This is Fig 14.5 in Goodfellow

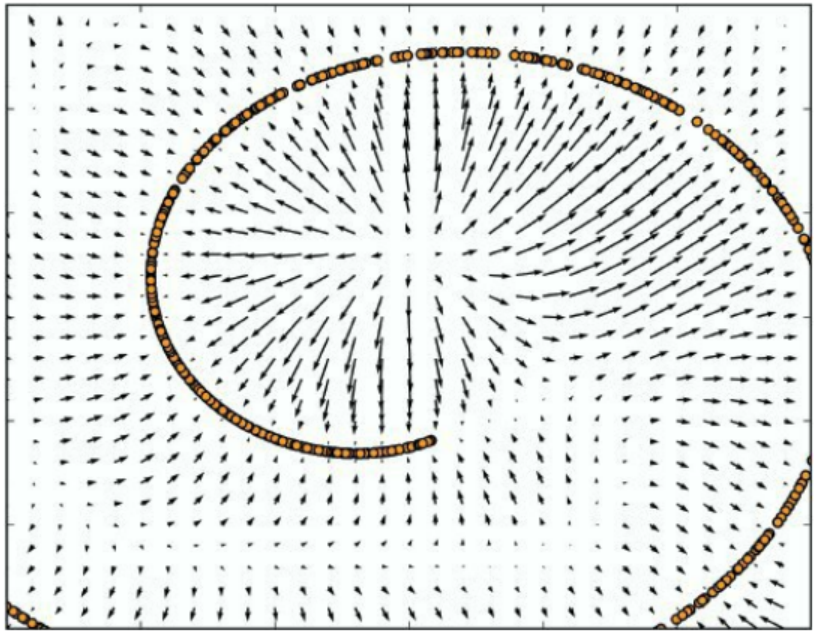


Figure 14.5: Vector field learned by a denoising

Goodfellow chapter 14.5 Denoising autoencoders contd

- Now use stochastics to 'push' the observations away from the manifold. The denoising function will have a gradient to push it back, and this gradient can be learned and will tell us something about the function

Goodfellow 14.6 Learning Manifolds with autoencoders

- As always when we learn with autoencoders, we have two objectives
- learn the best representation of data
- respect the restrictions of the model and not just learn the identity function
- Imagine the following example, using PCA to reduce the die

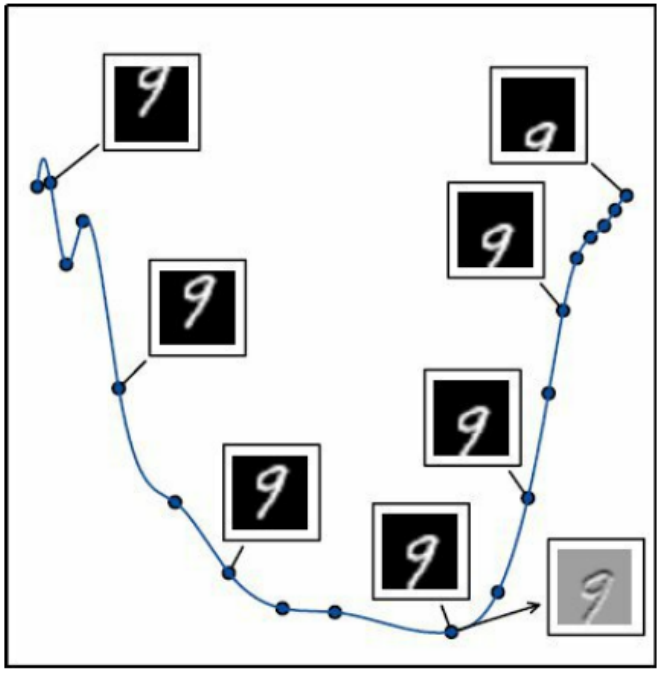


Figure 14.6: An illustration of the concept of a

Goodfellow 14.6 Learning Manifolds with autoencoders contd

- Transform pictures to PCA two-dimensions
- If the transformation model is good, pictures with the same label will be positioned around a one-dimensional manifold in the two-dimensional space
- Goodfellow has an interpretation of the tangents to the manifold that is difficult to understand
- Goodfellows point: If we can learn an encoder that is insensitive to noise we can capture the manifold structure of the data. The two things are the same. Difficult to understand, but of course if we push observations away from the manifold and they do not come back, we cannot learn the manifold. NOISE is key to autoencoders
- The projection of a point onto a manifold is sometimes referred to as an embedding

Goodfellow 14.7 Contractive autoencoders

- Intuitively - limit the size of the Jacobian matrix within the unit circle
- If the 'budget' for the total size of derivatives is small, we are forced to learn a few derivatives that will really drive a good fit

Goodfellow 14.8 Predictive Sparse Decomposition (PSD)

- PSD is an alternative algorithm which minimize $||x - g(h)||^2 + \lambda|h_1| + \gamma||h - f(x)||^2$
- Has computational benefits
- Can be stacked

Goodfellow 14.9 Applications of autoencoders

- Can reduce dimensionality with less reconstruction error than PCA
- Dimensionality reduction can make search more efficient - if done right
- Database application: reduce to low-dimensionality binary vectors that are keys in a hash table
- Binary values generated by sigmoids that return values close to either 0 or 1
- While Goodfellow talks about info retrieval, the course mostly talks about denoising autoencoders as part of generative AI

“

Goodfellow chapter 20

Goodfellow 20.10.3 Variational autoencoders

- G starts with generative models
- A discriminative model is a model of the conditional probability of an unknown variable Y given an observed variable X p(Y|X)
- A generative model is a model of the joint probability of an unobserved variable Y and an observed variable X p(X,Y)

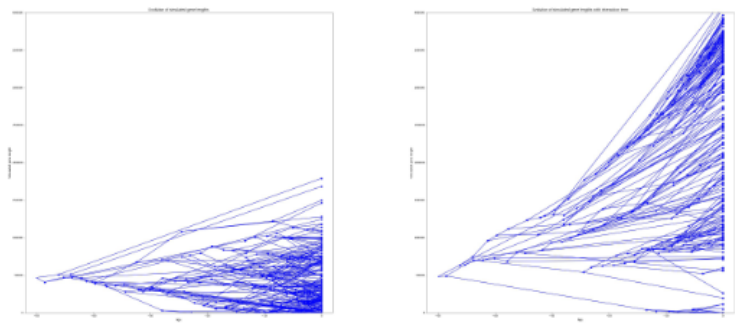


Figure 2.2: Evolution of simulated gene lengths without (left panel) and with (right panel) interaction term

4

Goodfellow 20.10.3 Variational autoencoders contd

- Not to be confused with generative AI
- Known from probabilistic graphical models in Models for Complex Systems. If X, today's observed gene length, is a function of Y, the unobserved gene length of previous generations, what do we expect Y to be

Goodfellow 20.10.3 Variational autoencoders contd

- Now apply this concept to an autoencoder. Assume we have PCA as a linear function that reduces dimensionality, it could also be a non-linear function. Assume we have MNIST images in 784 dimensions
- In an denoising autoencoder, we would add noise to the variables before dimensionality reduction, then encode them in a lower dimension, then project them back into full dimension and calculate our model error
- In a variational encoder, we map points in 784-dimensional space to probability distributions in lower-dimensional space. Let us say that an image represents 9 in 784-dimensions and a latent space has 25 dimensions. Assume that a an autoencoder would normally map the image to the vector (100,...,124)

Goodfellow 20.10.3 Variational autoencoders contd

- Instead of mapping from a 784-dimensional vector to a 25-dimensional vector, the variational encoder maps to a probability distribution over 25 points. If the encoder would normally map to the point $\{x_1, ..., X_{25}\} = (100, ..., 124)$, the variational encoder maps to a probability distribution of the joint probabilities of $\{x_1, ..., x_{25}\}$. The probability distribution is limited to being a Gaussian with the mean $\mu = (100, ..., 124)$ and a given variance.
- In order to decode, the denoising autoencoder would take the projection of the corrupted input into 25-dimensional space and project it back up to 784-dimensional space.
- The VAE instead samples from the probability distribution, projects back into 784-dimensional space and measures the error relative to the actual input, then trains the model

Goodfellow 20.10.3 Training a variational encoder

- The VAE knows that it is dealing with probability distributions and therefore needs to measure error in a certain way. One option is Monte Carlo but that requires numerous simulations
- Kullback - Leibler has the advantage of providing a closed-form solution
- The output of the decoder will follow a Gaussian due to the restriction on the encoder
- Learn the mean and standard deviation from the actual values
- Calculate KL which is a measure of how much two Gaussians deviate from one another - this now becomes the error function to be minimized

Goodfellow 20.10.3 Benefits of a variational encoder

- Yet another way of forcing the autoencoder to learn
- Intuition of learning probability distributions rather than point observations
- Goodfellow mentions that a VAE has learned dimensions with a high degree of intuition (one is the angle of a face the second dimension is the facial expression) but not clear if this could only be learned by a VAE

Goodfellow 20.10.4 Generative Adversarial Network (GAN)

- Begin with Joseph Rocca `https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4`
- Now generative in the true sense of the word - produce an image of a dog that incorporates everything we have learned about dogs without being a copy of any one specific dog
- For economists this would be an optimization exercise - produce the 'optimal dog'
- Due to MLs roots in computer science, this is an exercise built on learning probability distributions and sampling from them (the most likely dog)

Rocca GAN

- A picture of a dog of 784 pixels can therefore be seen as a sample from a probability distribution over 784 variables
- This probability distribution is unknown and very complex - for instance the variables are far from independent
- Maybe we can also conceptually see it as a graphical model - a random dog is first a sample of species and other characteristace, these hidden variables will determine the 784 variables
- Rocca first discusses general methods to calculate probability distributions (Inverse Transform Method)
- This still assumes that we know the distribution we are looking for in one dimension
- ... and even more so in 784-dimensions

Rocca GAN contd

- Instead, let us use a neural network to act as the probability distribution. From Inverse Transform Method, we retain the idea that the input to the network is a sample from a uniform probability distribution.
- The output from the network will be 784 pixel values which can be seen as a random variable in 784 dimensions representing an image
- How do we train the model?
- First idea would be generate a large number of samples and identify the actual distribution over the 784 variables
- The compare it to the distribution over a test set of actual photos
- The loss function could be Kullback Leibler measuring the difference between two probability distributions

Rocca GAN contd

- Rocca prefers another way of training the model. Let us take pictures of dogs that are samples from our probability distribution, mix them with an equal number of real pictures of dogs, sample from that mix and show the samples to people asking 'is this a real dog or a computer generated dog'
- We want people to be WRONG
- If the model is perfect, the test person cannot discriminate but knows its 50-50 so flips a coin each time and is right 50 per cent of the time
- We ignore the responses when people are shown real pictures
- If computer-generated pictures are classified correctly more than 50 per cent of the time, we update the model

Goodfellow GAN

- Obviously, this experiment is not practical - therefore let us use an ML-classification model trained to discriminate real dogs from ML-generated dogs
- if the model is right more than 50 per cent of the time, we update the neural network
- We train BOTH networks
- If the reward function for the discriminator is convex, the game will converge
- If both are neural networks there is no guarantee of convergence

Goodfellow GAN contd

- The ideal situation is that the generator learns very well, the discriminator is only able to predict 50-50 and is discarded
- No guarantee of convergence to this ideal outcome. Goodfellow discusses some noncovergent dynamics but does not exclude the case where the discriminator 'wins' - always being able to discriminate no matter how good the generated photos are
- Intuitively this is because both g and d are 'equally smart'. Imagine a situation where d can discriminate perfectly - it will then continue to deliver new training input and g will continue to learn

Goodfellow GAN contd

- GANs can even learn to never show their exact training data by assigning it minus infinity probability
- LAPGAN is similar to the idea discussed above that a 'random picture' should be seen as a probabilistic graphical model
- LAPGAN iteratively assigns conditional probabilities to (not labels but) low-resolution versions of an image and then iteratively add detail. The model is trained on various levels of resolution as well. It is not clear whether the adversary also intervenes at different levels of resolution

4. Recurrent models. Transformers

Goldberg chapter 10

Goldberg chapter 10

- The topic changes on two levels: recurrent models and Natural Language Processing (NLP) instead of image analysis
- Before the models, consider the input
- Text and corpus. N different words in corpus
- Words are encoded into vectors. Simplest encoding is one-hot encoding in N-dimensional vectors
- Combining words into sentences can be done by concatenation, which creates long vectors, or addition (such as bag-of-words BOW vectors) which destroys information about sequence
- Convolutional Bag of Words (CBOW) adds words but only within a subsection of the text

General considerations on word vectors

- RNNs that are the topic of Goldberg chapter 10 take individual words as inputs in the form of sequences of vectors.
- These vectors can both be one-hot and more complex encodings
- Two implementations are word2vec by Mikolov (2013) and GLOVE (Global Vectors for Word Representations) by Pennington et al (2014)
- Let us consider the encodings for a minute before we go further into language models

General considerations on word vectors contd

- My own naive version of word embeddings
- Assume a corpus of N unique words
- Define a neighborhood radius of k words
- For each unique word W, scan the neighbors in the radius of r words before and r words after
- Define symmetric weights of (2r-1), highest the closer to the word
- For each W, calculate an N-element vector of proximity scores of the other words. W itself may appear since a word can be its own neighbor. We now have an (NxN)-matrix of proximity scores, some of which are zero

General considerations on word vectors contd

- Transform the matrix of proximity scores into a matrix of distances. Diagonal values are coerced to zero. Zero-values are set to an arbitrary maximum distance. Relative proximities are transformed to relative distances. Call the matrix D_{actual}
- Now set a dimension value. Any set of relative distances between N words can be represented accurately in (N-1) dimensions, but the trick is to set dimensionality as much lower than N, e.g., K=512 for a whole language
- Initialise the model by randomly assign K-dimensional vectors to each of the N words. Calculate an N-dimensional matrix with zero diagonal showing Euclidean distances in K-dimensional space, call it D_{model}.
- Calculate the average distance in D_{model}. Normalise D_{actual} to have the same average distance and call it D_{as} for 'actual standardised'

General considerations on word vectors contd

- Both matrices are obviously symmetrical. Define a loss function of the squared distances between similar elements $L = \Sigma_{i=1}^{N}\Sigma_{j=1}^{N}|D_{as}(i,j) - D_{model}(ij)|^2$
- The vector coordinates are continuous in K-dimensional space, so L is a continuous function of N*K-parameters, with $K << N$
- The coordinates can be set so that L is minimized, using gradient descent
- The result will be meaningfuld embeddings of N words in a vector space that is a lot smaller than N dimensions

General considerations on word vectors contd

- This is already a machine learning model, before we apply recurrent networks or transformers
- The low dimensionality means that we can now concatenate rather than add
- The curriculum mentions remarkable interpretations of the vectors
- king - man + woman = queen
- Let us imagine the vector products. Two orthogonal words with vector product 0 have little relation, e.g., fish and bicycle
- Words with a high vector product have a high degree of relationship
- If a vector 'Biden' can be written as -1*'Trump' then the two words have a strong negative relationship

General considerations on word vectors contd

- This will become even more clear in transformer models that leverage these vector products of words strongly and de-emphasize the sequence
- So transformers are a breakthrough model in their own right but also they leverage the word embeddings in low-dimensional space in a different way than recurrent models

Recurrent models - Goldberg chapter 10

- Input vectors are x_t and the model also maintains a state vector s_t
- The state vector is the output of a neural network R that takes two inputs, $s_t = R(x_t, s_{t-1})$
- The prediction of the model at time t is the function of another neural network O which takes as its input the state vector $y_t = O(s_t)$
- R and O have the same weights and architecture at all times t
- The state vector integrates all words in the sequence up to that point
- Imagine the first word in the sequence. Ignore the state vector of the previous period, apply R to generate the state vector s_1 which will serve as input to O, a neural network that generates an output y $y_1 = O(R(x_1))$

Recurrent models - Goldberg chapter 10 contd

- In the next period, R takes an input of the next word x_2 and the state vector s_1. s_1 is an abstract representation of all the previous words that have been processed through the neural network R. R generates the new state vector s_2 which is fed into the neural network O.
- So s is neither a concatenation nor the sum of the previous words, it is an output vector from a neural network that serves as the input vector to another neural network. Its dimension is not necessarily K, it is not specified in the article
- The dimension of y is not specified either since the model can have several purposes, but in a labelled dataset y is in the format of the label

Recurrent models - Goldberg chapter 10 contd

- Goldbergs description of the use cases is confusing.
- The first use case is what could be called 'static translation' which he refers to as training the model only on the final output sequence against the final label sequence. He calls this training the model as an 'acceptor'.
- Transducer trains the model on a series of local loss functions, first for the first word, then for the two first etc, so it is a more real simulation of real-time translation.
- A special case of this is then language modelling, prediction the i'th word based on the (i-1) first words in the sequence, something that could already be done with ngrams.
- Then the final case - encoder decoder-based machine translation

Challenges presented by simple RNN and how to deal with them

- Includes context but only before the word, not after
- This can be solved by bi-RNN. Flip the sequence of x, maintain two separate RNNs R^f, O^f and R^b, O^b. The output at time t is the concatenation of the two outputs O_t^f, O_t^b. It is not stated how this concatenated vector is then processed into a label prediction. The model is then trained on the labels
- Another problem is vanishing gradient since the models obviously become very long
- This is solved in two different ways, Long Short-Term Memory and Gated Recurrent Unit
- The models are 15 years a part in time and not clear what the exact difference is

Challenges presented by simple RNN and how to deal with them contd

- LSTM has a more elaborate data structure in the place of the state vector: a state vector plus a memory unit that stabilizes the input in a way that also stabilizes the gradient. It is selectively decided by a gated unit whether to update the memory unit
- Intuitively, think of the memory unit as a canon of language that determines statements by a group of people at a given time. Only if certain logical conditions are met is the canon updated.
- This update function is performed at each time t
- The update function is then trained

Challenges presented by simple RNN and how to deal with them contd

- Interpretation of update function: the importance of the context relative to the current input
- Gated Recurrent Unit solved this problem in a different way 15 years later
- What is the difference between GRU and LSTM? GRU has an update gate that cuts off the text at the appropriate place, LSTM has a forget gate that learns to forget the appropriate words not necessarily as a function of the sequence

Challenges presented by simple RNN and how to deal with them contd

- Computational efficiency supported by the fact that R maintains the same weights at all positions and O maintains the same weights at all positions
- Difficult to parallelize due to sequential nature

“

Vaswani

Vaswani Attention is all you need

- Attention mechanisms had been used as a supplement in previous models but Vaswani was the first to have attention as the primary prediction mechanism
- From the outset, V seems focussed on use cases such as real-time translation and therefore the pure attention-based model is adapted in a number of ways in order to 'prevent left-ward information flow in order to preserve to auto-regressive property'
- Leftward information flow can actually be helpful - remember bi-RNN
- Therefore, this presentation starts with a pure attention-based model, then moves on to the Vaswani implementation
- Assume we are training the translation into Danish of a novel by Raymond Chandler

What is attention

- Words are represented by embedded vectors as developed by word2vec Mikolov (2013) or GLOVE Pennington (2013)
- This representation already represents a lot of information
- Vaswani had 512-dimension vectors and preserves this dimension throughout
- Now process the next word in the input sequence x_t into an encoded-decoded output y_t
- In the purest attention-based model, we take the vector product of x_t with every other word in the text, producing a (n-1) vector of products

What is attention contd

- Words that appear k times result in k vector products
- We process this (n-1) set of vector products through a feed-forward neural network where the layers are 512 nodes broad
- The final activation function is a 512-dimension softmax that outputs a word, y_t
- Assume we get the prediction of x_t wrong - x_t was 'fish' and our model predicted either 'en fisk' (noun) or 'at fiske' (verb)
- It turns out that the correct translation was 'den nye medfange' (the new inmate)

How are attention-based models trained

- The model learns to make this new prediction by updating the weight of the vector product between x_t (fish) and the word 'prison' which also occurs in the text
- In the model, fish 'attends to' prison. We learn that there is a relationship between fish and prison that affects the in-context translation of fish
- The relationship between prison and fish can be long range in the text. It may be in a previous chapter that we learned the situation is set in a prison and therefore 'fish' means 'new inmate'
- In the general corpus, 'fish' and 'prison' were almost orthogonal - they had very little proximity, but the model is able to learn a context-specific relationship as well as leverage existing relationships over long ranges in the text

What are the benefits of attention-based models

- More computationally efficient both in training and prediction
- Better BLEU-scores (Bilingual Evaluation Understudy)

Why are attention-based models more computationally efficient

- The benefits presented by Vaswani comprise all his adjustments to the model, but in the basic case
- In principle, attention-based models are $O(n^2)$ since all words are processed against each other, which obviously is not very fast. However, the Vaswani implements a number of things that make the model fast
- Unlike a recurrent model the calculations can be parallelised
- Attention-based models rely on vector and matrix representations that in practice can be quite fast on modern computers

Attention heads

- All the more so since Vaswani splits vector multiplications into vector products of 64-element subsections of the 512-element word vectors
- Such a subsection is called an 'attention head'
- Even the heads turn out to be very significant and can have interpretations in terms of specific heads 'exhibit behavior related to syntactic or semantic structure of the sentences'

Vaswani implementation - positional encodings

- Vaswani implements an encoder - decoder model for translation from English to French
- Obviously, we cannot be completely agnostic as to the relative position of words
- We therefore 'something' to the input words at the position i
- In the simplest form, we add the integer 1 to all coordinates of the 512-element vector representing the first word in the text, we add the integer 2 to all coordinates of the second word and so on
- When we calculate the vector products and learn their weights, a word in the beginning of the text would attend differently to the various other words than the same word being input later in the text
- There are some words which can determine the meaning of the entire text, for instance the first sentence of a verdict

Vaswani implementation - positional encodings

- However, this may not be the best positional encoding
- Imagine instead that every sentence in the language has exactly 10 words. We could then add $(i-1)|10 * w$ to every coordinate of every word to reflect the position in the sentence. That way, the word 'fish' on the first position in a sentence would attend to 'prison' differently than the word 'fish' in the fifth position
- Vaswani applies a sinus and cosinus function to get the same effect but in a non-linear way
- He applies differently to different coordinates - again a reflection of the fact that different heads reflect different dimensions
- So the relative position of x_i in a cycle is input to x_i before being encoded into the model

Vaswani implementation - overall architecture

- Once the positional encodings have been added, the 512-element vector representing x_i is input into the model
- There are 6 layers each of which contain
- attention layer
- feed-forward neural network
- Residual layer
- The output of the attention layer is input into the feed-forward layer the output of which is added to the word via the residual layer, so each individual layer updates the word rather than transforms it

Vaswani implementation - queries, keys, values and output

- So far we have described attention mechanisms as simple vector products to which weights were applied
- Vaswani's implementation introduces 4 data structures: query, key, value and output. Q, K and V have separate weights and are input into a softmax function to produce 512-dimensional probabilities
- Sequencing in the model is managed by distinguishing between encoder-decoding attention, where the queries come from the previous decoder layer and self-attention where queries and keys come from the same layer
- The self-attention layers mask out words to the right of i to prevent leftward information flow and preserve the autoregressive nature of the model
- The output of the decoder is probability distribution over 512 dimensions which produces the prediction $y_t = f(x_t)$

“

YLC

YLC Self-supervised Learning

- The article introduces the concept of self-supervised learning (SSL) and provides a rationale for denoising autoencoder, GAN and variational autoencoder that we have already encountered
- Call it SSL instead of unsupervised
- First argument for SSL: There is much more unlabelled than labelled data in the world
- Second argument for SSL: some use cases have insufficient labelled data, for instance LLMs for languages with few native speakers
- Wider argument: machines must learn common sense, i.e., learn beyond a finite set of instructions (common sense)
- SSL is predictive learning: predict information that is already there

YLC Self-supervised Learning in images contd

- SSL uses the structure of the data itself and image SSL can use a variety of supervisory signals across video and audio (for instance, what should happen in the picture the moment before we hear a gun go off)
- NLP has a logical case for SSL: CLOZE. Since that use case is straightforward, YLC concentrates on the image use case
- Text is finite so we can have a predictive model for a text that returns a probability distribution over a well-defined domain or event space
- In images, event space is not finite, there is an infinte amount of video frames that can follow from a given frame in a video
- Therefore we cannot return an explicit probability distribution so we need to do something else. This is the same conversation as for denoising autoencoders and variational autoencoders

YLC Self-supervised Learning in images contd

- Use case: We have a dataset of images. We want to perform SSL by blocking out part of the dataset, e.g., a patch of an image and see if we can predict that patch from the rest of the image, y' predicted from y. We have the correct patch to learn from
- YLC takes a different approach going back to Shannon's original information theory and wants to use the energy function to indicate the extent to which y' is different from y
- The energy function is a function of two elements that represents the amount of information, if y' is similar to y there is no new information so energy is low, if it is wildly different there is a lot of information so energy is high
- He then builds up a 'Siamese twin'-network of two separate neural networks, one if fed y as input and the other is fed y'.

YLC Self-supervised Learning in images contd

- It is now easy to train the network to return a low energy for pictures that are similar
- The use case for pictures that are similar are pictures where B is generated by adding a small amount of noise to A. Knowing that B is generated this way constitutes the 'label' for training the model on pairs of (A,B)
- It is straightforward to train the model to return low energy for pairs of the type (A,A+noise). Maybe extend the model with measures such as Euclidean distance to cover other forms of similarity
- No systematic way of ensuring that all different pictures return a high energy. Tolstoj: 'all similar pictures look alike, all different pictures are different in their own way'.
- Avoid a 'collapse' of the model where it returns low energy for all combinations

YLC denoising autoencoders, GANs and VAEs are use cases of SSL

- Thankfully, part of the response lies in the models we have already learned.
- Denoising autoencoder learns from adding noise to images
- GAN is only part of the solution - GAN can learn to generate similarity, i.e., take a training set of images, delete a patch, generate something that is not identical to the patch but which cannot be distinguished from the patch by even the smartest discriminator
- Similar to the above arguments, it is not clear GANs protect us from the 'worst Type I error' - an image that according to common sense is wildly different but the generator outputs believing that it is similar
- Variational autoencoders can also learn from a set of images and output images that are not similar but difficult to distinguish

“

Dor

Dor: Learning thematic similarity in text

- Example of SSL - here contrastive learning
- Topic is text not images
- Train an unlabelled model
- This is SSL so the unlabelled data holds a lot of structure and information
- Text is divided into sections
- Sentences in the same section are more thematically similar than two sentences from two different sections
- Generate a dataset of triplets: A is randomly selected sentence, B is from the same section, C is from another section.
- In a subsequent iteration, to make the experiment more challenging, C is from an adjacent section

Dor: Learning thematic similarity in text contd

- Words are represented by GloVe embeddings so the simplest approach would be to calculate 'the average word' in each sentence and calculate Euclidean distances in 512-dimensional space
- Dor instead defines a neural network and feeds in the sentences as concatenated word embeddings
- The network outputs a position in K-dimensional space, K is not specified
- Distances can then be calculated and a loss-function. The simplest imaginable loss function would be the $|A - B| - |A - C|$ so if the distance between A and B is a lot smaller than the distance between A and C, loss is minimized
- Dor instead defines something more complicated

Dor: Learning thematic similarity in text contd

- The model tests well and can be applied to for instance defining a pivotal sentence then extracting all sentences relating to the same theme from various texts, something that is done, e.g., by ChatGPT although probably applying a different technique since the Dor article is from (2018)
- We leveraged structure in the data, at no time did we read the article and label their degree of thematic difference

5. In-context learning. Fairness

“

Brown

Brown: LLMs are few-shot learners

- Definitions:
  - Fine-tuning
  - Few-shot
  - One-shot
  - Zero-shot

Brown contd

- Learning in models appears to scale continuously as a function of number of parameters and size of training data set but for very large models (GPT-3 has 175 billion parameters and is being compared to models with 125 MILLION parameters) the ability to meta-learn seems to appear
- Zero-shot performance grows steadily but few-shot performance increases more rapidly, demonstrating that large models are more efficient in-context learners
- Learning occurs because the model is autoregressive, no weights are being updated, the model autoregressively projects the translation of the next token x_t and if questions appear in the input they are responded to
- Yova: this is next-token prediction like other autoregressive models

“

Piqueras Søgaard

Piqueras Søgaard are MULTI-lingual models fair across languages

- Difficult to understand use case and the element of unfairness being discussed
- The simplest use case: a language model performs cloze-tasks and is compared to responses to the same cloze-tasks performed by different groups MN, MNN, FN, FNN.
- We assume that the model is able to translate correctly, so there must be a BLEU-score hidden somewhere under the numbers
- We also assume that the test persons are able to provide relevant answers to the cloze-tests
- This is not really demonstrated in the article

Piqueras Søgaard are MULTI-lingual models fair across languages contd

- In a fair world, the degree of overlap between the responses by the model and the human responses would be $\alpha + \delta$ for MN and FN and α for MNN and FNN, $\alpha, \delta \in [0, 1]$ and $(\alpha + \delta) \in [0, 1]$ so the model would simulate male speech just as much as it would simulate female speech and there would be a 'natural' difference between native and non-native speakers
- Unfairness arises if the degree of overlap differs significantly between M and N or if the δ between native and non-native speakers is somehow 'too large'
- This concept of unfairness could be tested in a mono-lingual model.
- The model is unfair if it produces speech that is more similar to certain demographic groups than to others

Piqueras Søgaard are MULTI-lingual models fair across languages contd

- These groups may find it difficult to interact with applications that use that particular application or more generally feel disenfranchised by text generated by the model
- This concept of unfairness is not dealt with explicitly by the article
- The focus of the article is 'given we have a vector of differences in overlap between the four demographics for the model in a given language, what if this vector differs from the vector of differences relating to another the same model in another language'
- The use case: models are trained more on English-language data than other data, so the vector of differences may be smaller for English than for other use cases then that is unfair - that underrepresented languages will see larger differences across demographics

Piqueras Søgaard are MULTI-lingual models fair across languages contd

- He then adds a dimension of comparison across models but does not really conclude

mBERT

P@1	EN	ES	DE	FR	
MN	13.3	12.7	11.3	10.7	12.0 (1.0)
FN	13.3	12.0	15.3	8.0	12.2 (2.7)
MNN	12.7	12.4	11.4	3.6	10.0 (3.8)
FNN	13.3	10.0	5.6	6.9	9.0 (3.0)
	13.2 (0.3)	11.8 (1.1)	10.8 (3.5)	7.3 (2.5)	$P@1(\sigma_{gl})$

mBERT

P@5	EN	ES	DE	FR	
MN	30.7	26.7	22.0	24.0	25.9 (3.3)
FN	32.0	18.7	24.7	22.0	24.4 (4.9)
MNN	34.0	25.9	12.1	15.0	21.8 (8.7)
FNN	32.7	25.3	16.3	16.3	22.7 (6.9)
	32.3 (1.2)	24.2 (3.1)	18.8 (4.9)	19.3 (3.8)	$P@5(\sigma_{gl})$

- Assume EN is the perfect vector (actually quite close) then DE and FR have much too small values for MNN and FNN and are too disparate for MN and FN
- This would indicate mBERT is unfair
- Disregard absolute values of vectors which are not intuitive

Jørgensen Søgaard Rawlsian AI fairness loopholes

- Rawlsian fairness: when selecting across various distributions of benefits, select the distribution that has the highest benefit for the group that is worst off - EVEN IF THIS ABSOLUTE MAXIMUM FOR THE WORST OFF MEANS ACCEPTING A HIGHER RELATIVE DISPARITY
- Rawls also has a comment about it being attached to offices open to all, so the fair condition must not be administered by a dictator
- Now apply the notion of Rawlsian fairness to the introduction of AI-NLP speech recognition in a society where complete equality prevails
- It is clear that introduction of this new technology would not preserve perfect equality

Jørgensen Søgaard Rawlsian AI fairness loopholes contd

- Subgroup Test Ballooning: a certain demographic is selected as pilot group in development. The test therefore optimizes relative to the requirements of this demographic, increases followership among that demographic but becomes less suited to other demographics
- 'The technology will help others down the line' - but what if it does not?
- Snapshot-representative evaluation: the technology is applied fairly to its current users but would not be fair if applied more broadly
- Therefore, one option could be to require of the technology providers that they maintain approximate equality when rolling out the new technology. This would require them to allocate specific ressources to adapting the technology to the needs of marginal demographics

Jørgensen Søgaard Rawlsian AI fairness loopholes contd

- In fact it would be possible to go one step further: use the technology actively to solve existing inequalities (which apparently still exist in this perfectly equal society)
- Nielsen's definition of equality: each person will have the right to an equal share of income and wealth

Søgaard slides

Søgaard slides

- Subgroup test ballooning: 'the technology will be fair when scaled up'. What if it is never scaled up?
- Statistical parity: The chance of being healthy for a man must be equal to the chance of being healthy for a woman
  $p(d = 1|g = m) = p(d = 1|g = f)$
- Performance parity: The chance of having a high income must be the same for a healthy man as for a healthy woman
  $p(y = 1|d = 1, g = m) = p(y = 1|d = 1, g = f)$
- Accuracy equality: The proportion of rich men must be the same as the proportion of rich women
  $p(y = 1, g = m) = p(y = 1, g = f)$
- It is clear that we can have situations that are fair under one metric but not under the other. That is the point of having several metrics

Søgaard slides

- Hedden (2021) has constructed an example that had no statistical parity, no performance parity and no accuracy equality
- Søgaard challenges the example saying that Hedden evaluated a finite set of examples and fairness applies to entire distributions
- My comment would be that the example does not prove the previous comment about incompatibility of fairness metrics

6. Vision transformers

“

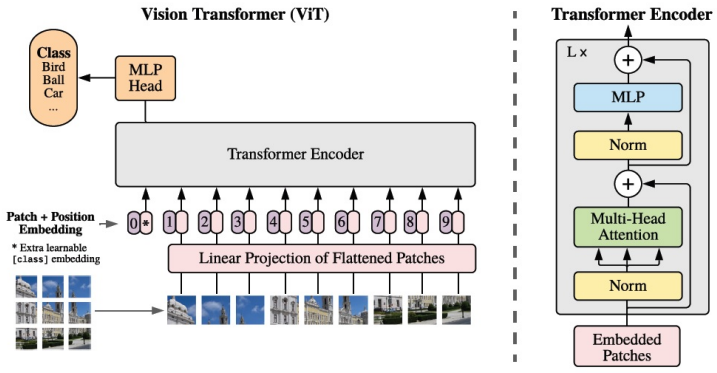
Dosovitskiy

Dosovitskyi - Transformers for image recognition

- Transformers were initially developed for NLP but are here applied to image analysis
- First use case is to train classification on a labelled data set
- The naive approach - calculate attention pixel to pixel - is not realistic so the picture is divided into patches
- Patches are embedded. D does not really describe how but obviously patches can have Euclidean distance
- The point of embedding: transform input data from a high-dimensional space to a much smaller embedded space

Dosovitskyi - Transformers for image recognition

- Positional encoding is added to the embeddings
- The model reads the patch embeddings prepended by a label
- Similar to Vaswani each layer has a self-attention layer and a feed-forward layer
- The model ends in a feed-forward layer, an activation and a class prediction
- Weights are trained by supervised learning



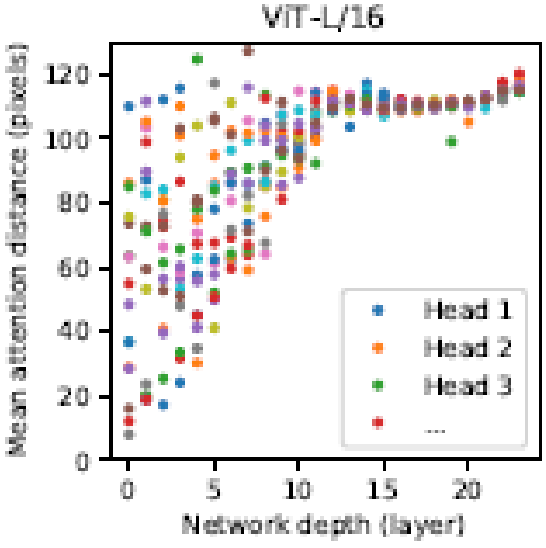
Dosovitskyi - general observations on performance

- Same conclusions as for NLP-applications of transformers: they are able to outperform other models
- D compares with convolutional image classification models
- D applies a pre-trained model
- Transformer models are rarely less good than convolutional models
- Transformer models outperform significantly when pre-trained on very large datasets (14 million images and above)

Dosovitskyi - inductive bias

- Inductive bias - the term is used a lot in ML and although bias is a negative, the right inductive bias in a model is a good thing

- CNNs have inductive bias in the form of spatial invariance, translation invariance or translational equivariance - this is the ability to recognize a shape irrespective of its position in the picture



Dosovitskyi section 4.5

- Transformer models do not have spatial invariance, translation invariance or translational equivariance in principle but seem to learn it
- D interprets this across heads rather than across input vectors
- The first layers learn both longer term and local relationships
- The deep layers learn longer term relationships

Dosovitskyi - performance

- The models developed by D are labelled ViT
- Reading the table downwards compares performance for the same model across various training sets
- Reading the table left-to-right compares performance across various models for the same training set
- Transformer performance is good especially for very large datasets
- What about detection and segmentation

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

“

Burt

Burt the Laplacian pyramid

- An old article from 1983.
- Pixels in images are highly correlated with their neighbors - can we eliminate redundant information and extract the salient features?
- Can a downscaled version of a patch serve as a low-dimensional embedding of a patch to be input into a transformer model?
- Previously we have seen that pictures can be upscaled using maxpooling or averagepooling
- We can also generate a pixel value by applying a kernel to a neighborhood of the pixel and setting the value of the pixel in the next layer of the neural network to the resulting matrix product. Might not extract salient feature - but that happens when we apply learning
- Burt shows how mathematical functions can be used to downscale and upscale images

Burt the Laplacian pyramid contd

- Generate a picture where pixels are averaged across a given interval k. Subtract that image from the original image, you then have a transformed image showing the salient features
- Mathematically, this is equivalent to taking the Laplacian operator, i.e., the second derivative $\nabla \cdot \nabla$ so it is called the Laplacian operator
- He demonstrates that empirically, the transformed image has close to the maximum entropy since entropy is surprise
- Quantizing, i.e., selecting only every k'th pixel, will reduce entropy
- From Stone Information Theory: Entropy is surprise. Expecting a coin to be fair and then flipping 10 times heads maximizes entropy
- Entropy of a distribution is $H(x) = \sum_{i=1}^{n} \frac{1}{n} log \frac{1}{p(x_i)}$ but somehow Burt says entropy is $H_{Burt}(x) = -\sum_{i=1}^{255} f_i log_2 f(i)$ so the probability of a pixel taking on the shade x_i times the log of that probability

“

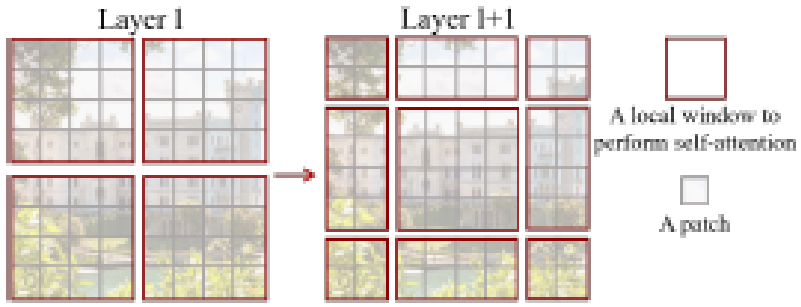
Ze Liu

Ze Liu Shifted windows

- Dosovitskiy introduced using transformers for images - only use case was classification
- Ze Liu uses shifted windows, improves accuracy and speed, covers all three use cases classification, detection and segmentation, even semantic segmentation
- However, detection and segmentation can only be done if inductive bias in the form of locational equivariance is included in the model
- Same problem as mentioned in the discussion of Dosovitskiy - that transformer models are in principle quadratic in time complexity. In the simplest form of the model, every token, pixel or patch must attend to every other token, pixel or patch

Ze Liu Shifted windows contd

- Segmentation into windows makes time complexity linear
- Between each layer, windows are then shifted in both dimensions by half the length of the quadratic window
- Instead of padding, marginal windows are combined with each other
- Shifting rather than sliding improves processor performance
- Within the same patch, the key-value set are shared and this increases efficiency



Ze Liu Impressive speed-accuracy tradeoff

- When comparing two models, compare FLOPS and accuracy for both

- SWIN-models are marginally better in the small training set and clearly better in the large training set

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384^2	307M	190.7G	27.3	76.5
DeiT-S [63]	224^2	22M	4.6G	940.4	79.8
DeiT-B [63]	224^2	86M	17.5G	292.3	81.8
DeiT-B [63]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [38]	384^2	388M	204.6G	-	84.4
R-152x4 [38]	480^2	937M	840.5G	-	85.4
ViT-B/16 [20]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [20]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [68] and a V100 GPU, following [63].

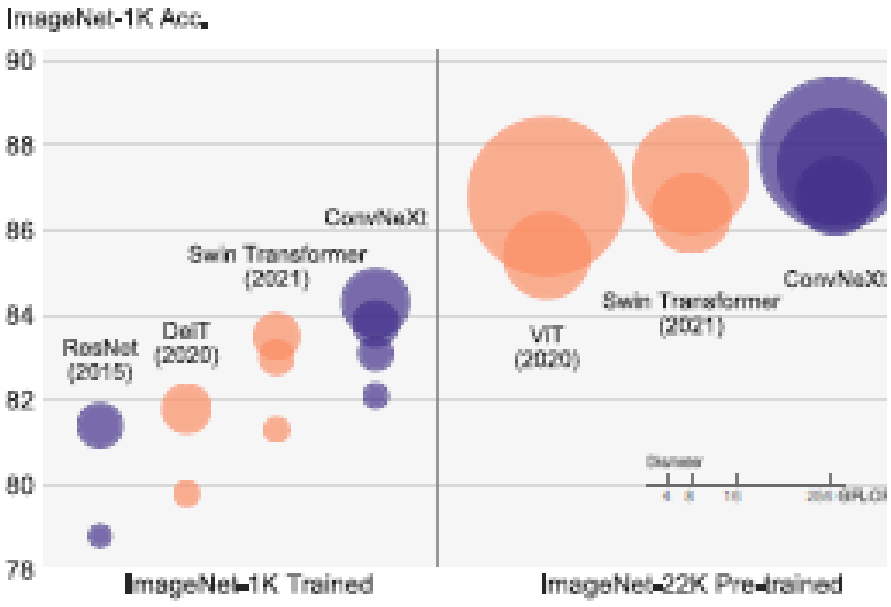
“

Zhang Liu

A ConvNet for the 2020s

- 2022-article which does not include transformers but which incorporates a number of other inventions
- Patchify stem = take the patching approach from transformers and combine it with the upscaling approach from CNNs
- Stage compute ratio
- ResNeXt design - patcher kun i den spatiale dimension ikke i farvedimensionen
- Inverted bottleneck: the hidden dimension is 4 times larger than the input
- Large kernel sizes
- New activation function (replacing ReLU with GELU)
- LayerNorm

ConvNeXt performance



model	image size	#param.	FLOPs	throughput (image / s)	IN-1K top-1 acc.
ImageNet-1K trained models					
• RegNetY-16G [54]	224^2	84M	16.0G	334.7	82.9
• EffNet-B7 [71]	600^2	66M	37.0G	55.1	84.3
• EffNetV2-L [72]	480^2	120M	53.0G	83.7	85.7
◦ DeiT-S [73]	224^2	22M	4.6G	978.5	79.8
◦ DeiT-B [73]	224^2	87M	17.6G	302.1	81.8
◦ Swin-T	224^2	28M	4.5G	757.9	81.3
• ConvNeXt-T	224^2	29M	4.5G	774.7	**82.1**
◦ Swin-S	224^2	50M	8.7G	436.7	83.0
• ConvNeXt-S	224^2	50M	8.7G	447.1	**83.1**
◦ Swin-B	224^2	88M	15.4G	286.6	83.5
• ConvNeXt-B	224^2	89M	15.4G	292.1	**83.8**
◦ Swin-B	384^2	88M	47.1G	85.1	84.5
• ConvNeXt-B	384^2	89M	45.0G	95.7	**85.1**
• ConvNeXt-L	224^2	198M	34.4G	146.8	**84.3**
• ConvNeXt-L	384^2	198M	101.0G	50.4	**85.5**
ImageNet-22K pre-trained models					
• R-101x3 [39]	384^2	388M	204.6G	-	84.4
• R-152x4 [39]	480^2	937M	840.5G	-	85.4
• EffNetV2-L [72]	480^2	120M	53.0G	83.7	86.8
• EffNetV2-XL [72]	480^2	208M	94.0G	56.5	87.3
◦ ViT-B/16 (☎) [67]	384^2	87M	55.5G	93.1	85.4
◦ ViT-L/16 (☎) [67]	384^2	305M	191.1G	28.5	86.8
• ConvNeXt-T	224^2	29M	4.5G	774.7	**82.9**
• ConvNeXt-T	384^2	29M	13.1G	282.8	**84.1**
• ConvNeXt-S	224^2	50M	8.7G	447.1	**84.6**
• ConvNeXt-S	384^2	50M	25.5G	163.5	**85.8**
◦ Swin-B	224^2	88M	15.4G	286.6	85.2
• ConvNeXt-B	224^2	89M	15.4G	292.1	**85.8**
◦ Swin-B	384^2	88M	47.0G	85.1	86.4
• ConvNeXt-B	384^2	89M	45.1G	95.7	**86.8**
◦ Swin-L	224^2	197M	34.5G	145.0	86.3
• ConvNeXt-L	224^2	198M	34.4G	146.8	**86.6**
◦ Swin-L	384^2	197M	103.9G	46.0	87.3
• ConvNeXt-L	384^2	198M	101.0G	50.4	**87.5**
• ConvNeXt-XL	224^2	350M	60.9G	89.3	**87.0**
• ConvNeXt-XL	384^2	350M	179.0G	30.2	**87.8**

ConvNeXt performance

- The real use case: detection and segmentation
- Outperforms transformer-based models

backbone	FLOPs	FPS	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Mask-RCNN 3× schedule								
○ Swin-T	267G	23.1	46.0	68.1	50.3	41.6	65.1	44.9
● ConvNeXt-T	262G	25.6	**46.2**	67.9	50.8	**41.7**	65.0	44.9
Cascade Mask-RCNN 3× schedule								
● ResNet-50	739G	16.2	46.3	64.3	50.5	40.1	61.7	43.4
● X101-32	819G	13.8	48.1	66.5	52.4	41.6	63.9	45.2
● X101-64	972G	12.6	48.3	66.4	52.3	41.7	64.0	45.1
○ Swin-T	745G	12.2	50.4	69.2	54.7	43.7	66.6	47.3
● ConvNeXt-T	741G	13.5	**50.4**	69.1	54.8	**43.7**	66.5	47.3
○ Swin-S	838G	11.4	51.9	70.7	56.3	45.0	68.2	48.8
● ConvNeXt-S	827G	12.0	**51.9**	70.8	56.5	**45.0**	68.4	49.1
○ Swin-B	982G	10.7	51.9	70.5	56.4	45.0	68.1	48.9
● ConvNeXt-B	964G	11.4	**52.7**	71.3	57.2	**45.6**	68.9	49.5
○ Swin-B‡	982G	10.7	53.0	71.8	57.5	45.8	69.4	49.7
● ConvNeXt-B‡	964G	11.5	**54.0**	73.1	58.8	**46.9**	70.6	51.3
○ Swin-L‡	1382G	9.2	53.9	72.4	58.8	46.7	70.1	50.8
● ConvNeXt-L‡	1354G	10.0	**54.8**	73.8	59.8	**47.6**	71.3	51.7
● ConvNeXt-XL‡	1898G	8.6	**55.2**	74.2	59.9	**47.7**	71.6	52.2

“

Kirilov

Kirilov Segment anything (2023)

- A new multi-purpose model
- The model does several things but a place to start is recapitulate that visual transformers performed well on classification but less well on detection and segmentation and could not stand alone for these two latter tasks
- The article describes an overall ML project and goes into less detail with the specific algorithms

Kirilov SAM What does it do?

- Take a verbal OR visual task
- Identify corresponding segment
- Zero-shot - tested on tasks that are different from training tasks
- Outperforms conventional segmentation algorithms such as RITM

Kirilov SAM - what are the requirements

- A task for requesting a specific segment
- - must take verbal or visual prompt
- A model for identifying and classifying segments
- - must perform better than other transformers on detection and segmentation (semantic segmentation)
- Must be able to handle ambiguity

Kirilov SAM - what are the requirements contd

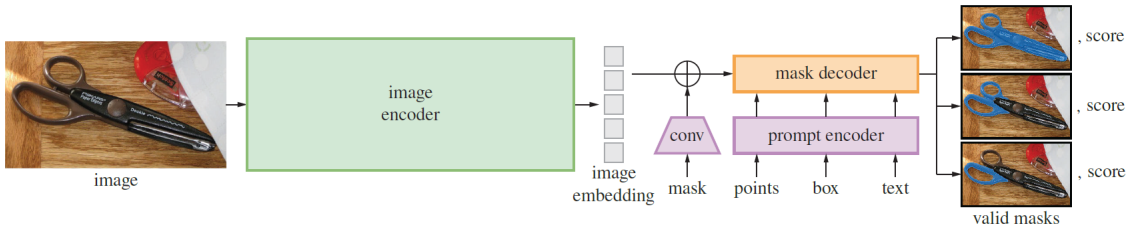
- A data engine to generate a training
- - unlike classification there was not a large labelled training set available
- - we had data sets pictures labelled 'red hat' but the segment representing the red hat was not identified
- Test zero-shot performance
- Run in amortized real time
- Analyze impact on responsible AI

Kirilov SAM task

- Task can be a point, set of points, box, mask or free-form text
- Given any must return at least one
- - Allowed to return more than one

Kirilov SAM model

- The model is 'built on vision transformer'
- The model will encode an image into an embedding vector
- A encoder - decoder reads the prompt, whether it is visual or verbal, encodes it in latent space and decodes it as a mask
- Attention is applied in two directions: prompt-to-image and vice versa
- Feed-forward network is applied
- The model generates one or more outcomes, if there are more outcomes each has a score and the outcome with the highest score is returned

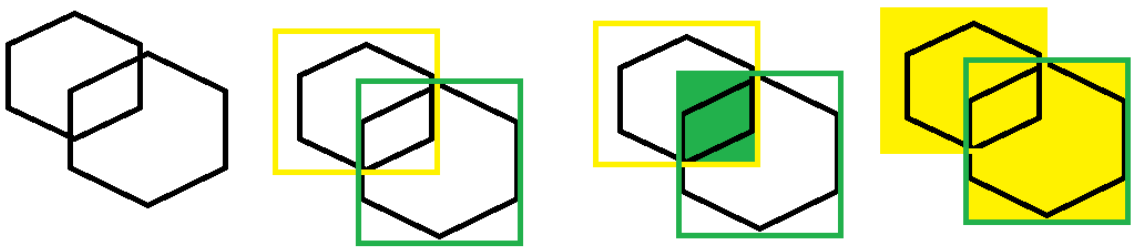


Kirilov SAM model handling ambiguity

- Model outputs three predictions (masks)
- Loss function and backpropagation is calculated as the lowest of the three losses
- - obviously, it will vary from iteration to iteration which mask has the lowest loss
- Confidence score is calculated for all three masks
- - calculated as IoU against training set (cf below)

Kirilov SAM IoU as a success function

- IoU = Intersection over Union. Take two masks, calculate their bounding boxes, measure their intersection and their unions and divide the two
- In the figure, the two hexagones are provided with a bounding box each, the intersection is marked in green and the union is marked in yellow. IoU is the area of the two divided by one another. The more similar the masks, the higher the IoU



Kirilov SAM Data engine

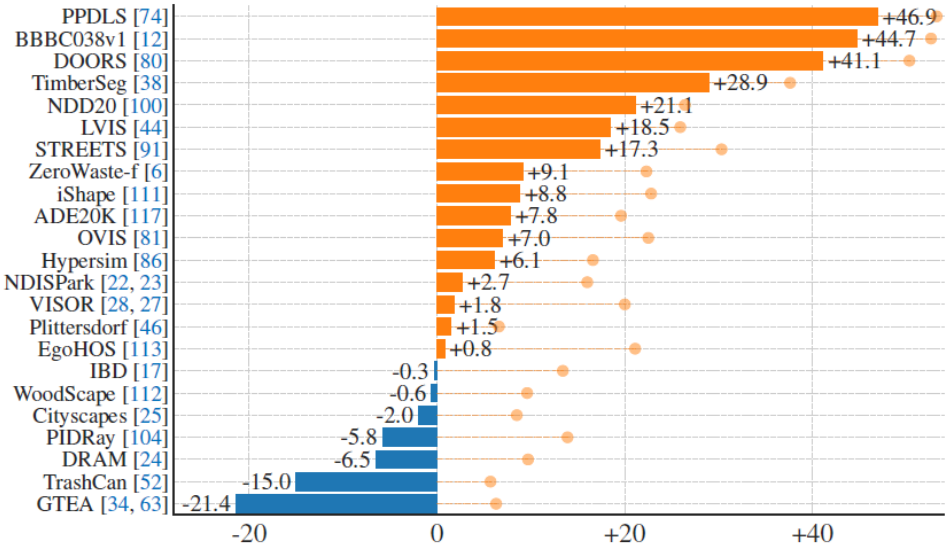
- Generates a dataset of 1.1 billion masks from 11 million pictures
- Assisted-manuel phase using human annotators and free-format
- Semi-automatic stage. Annotators are shown images with already identified masks and asked to identify ADDITIONAL masks
- Automatic stage. Procedure not clear but they now have a labelled dataset and can therefore both automatically identify masks and classify them
- 99.1 per cent generated in the automatic stage

Kirilov SAM Zero-shot performance

- The article uses the concept 'zero-shot transfer' which refers to an article by Radford et. al.
- Means that rather than fine-tune the model for a specific task, we can immediately apply it to tasks other than what it was trained for ('transfer to downstream tasks')
- Maps back to the overall objective of 'producing a valid mask from any prompt'
- Edge detection
- Segment anything, i.e., propose objects that were not readily evident in the picture
- Segment detected objects, i.e., instance segmentation (not clear what this means)
- Segment objects from free-form text

Kirilov SAM Zero-shot performance

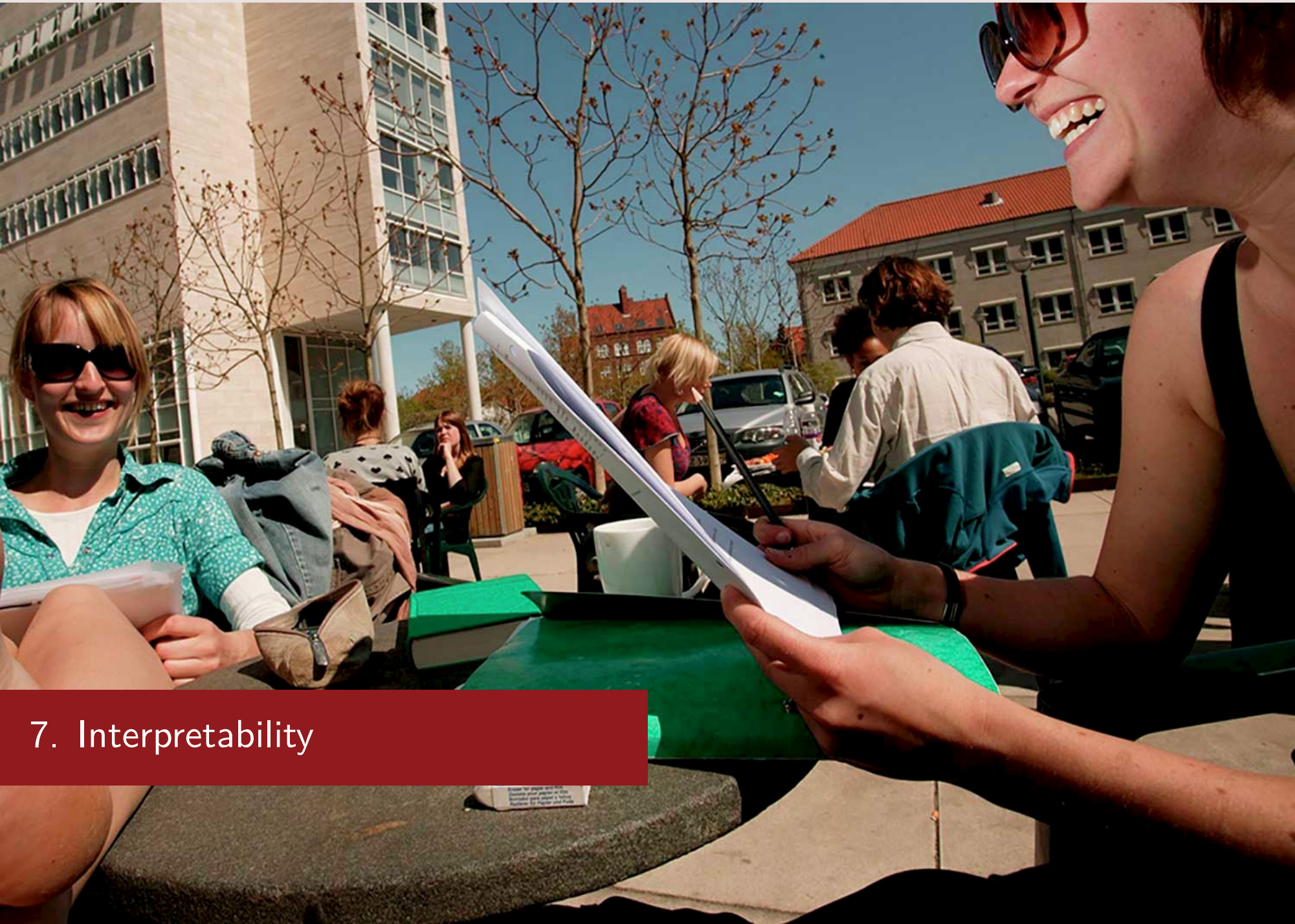
- SAM tested against existing segmentation algorithm RITM on zero-shot tasks on 23 different datasets
- Test results measured as difference in IoU
- Not clear that RITM could do all the things that SAM could do so would be a test of a SAM generated mask against a mask generated by RITM using a more traditional prompt process



Kirilov SAM edge detection and ambigous free-form prompts

- Edge detection on various datasets.
- SAM was never trained on edge detection
- SAM is outperformed on some datasets by much older specialized algorithms
- But the best use case is attributed to SAM
- Final use-case: ambigous free-text form 'beaver tooth grille' on a BMW

method	year	ODS	OIS	AP	R50
HED [108]	2015	.788	.808	.840	.923
EDETR [79]	2022	.840	.858	.896	.930
zero-shot transfer methods:					
Sobel filter	1968	.539	-	-	-
Canny [13]	1986	.600	.640	.580	-
Felz-Hutt [35]	2004	.610	.640	.560	-
SAM	2023	.768	.786	.794	.928



7. Interpretability

“

Feng

Feng Quizbowl experiment

- Existing literature has no agreed way of showing interpretation
- There generally considered to be three forms of interpretation:
- - visualizing uncertainty: the model predicts that white is more likely than black and the probability distribution is 65-35
- - humans generally have difficulty turning this form of interpretation into good decisions
- - highlighting important input features: 'The model advises that this person should not get a loan, and this is due to his high existing debt rather than his income'
- - Show relevant examples: The data shows that people with high existing debt levels often have difficulty repaying their loans, therefore we advise that this person should not get a loan

Feng Quizbowl experiment

- Beginners and pros are asked to play a game that balances accuracy and aggression
- They are shown ML-generated proposed answers and can not play trivial strategies
- Three different interpretations are provided
- - Evidence. Show the observations in the training data that are most directly related to that specific proposal
- - Highlight evidence. Similar to evidence, but in addition most significant words are highlighted
- - Highlight questions. Highlight the words in the questions that are most directly related to the proposed answer
- In general, interpretations lead to contestants being more succesful

Feng Quizbowl example

Question

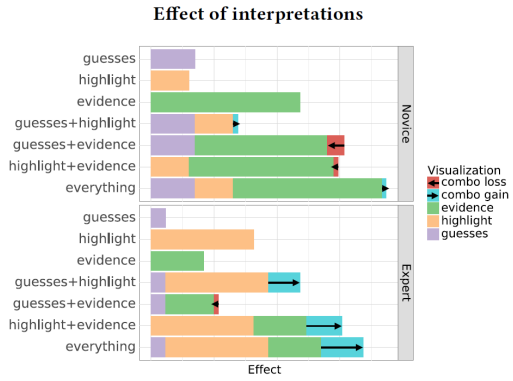
Its central basin is known as "the cuvette," and its navigable portion begins at Kisangani. It receives the Luapula and Lualaba Rivers, from whose effluence at Boyoma Falls this river receives its

Evidence

for **Congo River**

the Lualaba and the Chambeshi Rivers . It is navigable downstream from Kisangani , except for the area

Falls lies on this river , and after it reaches Kisangani , it is no longer called the Lualaba . This



“

Jiang

Jiang Two XAI cultures - studies of a deployed AI system

- Where Feng said that there were no agreed definitions of interpretability, Jiang refers to Doshi-Velez who defined iterpretability as 'ability to explain or to present in understandable terms to a human'
- In reality, AI-explanations have been targeted at AI professionals ('inmates running the asylum')
- Jiang broadens it somewhat to a case of XAI for other professionals such as compliance and legal functions in a bank evaluating a new AI for loan approval

Jiang Two XAI cultures - studies of a deployed AI system contd

- Specific examples: other professionals may understand ranked lists of feature importance or simple decision trees
- Compliance team focussed on non-discrimination
- Legal team focussed on covering all use cases
- Jiang finds that the relevant definition of interpretability is the ability to provide an explanation that supports one of three items:
- - Make decisions
- - Take actions
- - increase understanding of how AI models make recommendations

Jiang Two XAI cultures - recommendations

- Know the audience
- Help stakeholders develop a mental model of XAI
- Target the communication
- Incorporate insights from social science and domain expertise

“

Søgaard Explainable NLP

Søgaard explainable NLP

- First dichotomy: local vs global
- This dichotomy is already in literature and Søgaard maintains it
- Global is methods that rely on a sample of representative instances. This includes updating the model
- Local is methods that can provide explanations for individual instances
- What is the scope of local explanations?
- Local is concerned with explaining specific use cases. Why did this individual not get a loan?
- What is the scope of global explanations?
- Global is concerned with biases of the model on larger demographics

Local, gradient-based explanations

- A use-case for local explanations is gradient-based explanations
- The term gradient has previously been used in ADL in the sense the derivatives of the loss-function with regard to all the weights
- Here, it means the derivative of the output-function with regard to all the features
- Example: the label for the loan applicant was negative so he did not get a loan. The one feature that most affected this decision was his existing debt and no so much his income

Global attention-head based explanations

- A use-case for global explanations is attention-head pruning
- Pruning means modifying, let us be even more basic, assume the model is attention-based with no heads
- Then separate the vectors into segments, i.e., heads whose internal vector products have separate weights
- It has been shown above that these heads can have separate interpretations, one head is highly correlated with the semantic meaning
- Obviously, separating the vector products into heads requires a larger sample of instances
- So does modifying (pruning) these heads

Intrinsic vs post-hoc

- This is an existing distinction in literature and Søgaard discards it
- Intrinsic means outputting the explanation together with the model prediction
- Post-hoc means explaining using techniques that are orthogonal to the model
- Discarded because the term post-hoc had different meanings in local vs global approaches
- When we say post, do we mean post-training or post-inference?

Backward vs forward interpretability methods

- An interpretation method is said to be backward if it relies solely on quantities derived from one or more backward passes through the model.
- Since gradient-based updates are backward, gradient based methods are backward methods
- A model that relies on quantities from forward passes (and possibly ALSO backward passes) is said to be forward
- Approximating the model's output distribution is a forward method

Søgaard slides

Points made by Søgaard

- Not opaque but overwhelming and 'hard to summarize in a few words'
- Two kinds of opacity: Training opacity and inference opacity
- Reasons for opacity:
- - number of parameters
- - lack of grounding, i.e., decision making in models not similar to human decision making
- - Continuity - most human decision making processes are discrete
- - Lost training history: we do not know how we got to the model in the first place
- XAI is a summarization problem or an abstraction-faithfulness tradeoff

Points made by Søgaard

- Good explanations can ensure that we are right for the right reasons, e.g., do not correctly predict based on irrelevant but highly correlated parameters (doctor vs nurse predicted based on gender)

Other points made by Søgaard

- Taylor-approximation can support explainability
- Since a neural network is a function, assuming it is also continuous it can be approximated by a Taylor-polynomium
- In a given data point, the coefficients of the Taylor-polynomium will be numeric values
- Sum them together across variables and interpret

Brandl slides

Brandl What is a good explanation

- Two types: WHY something happened or HOW something happened
- Sparse
- Faithful
- Stability/sensitivity
- Plausability/efficiency

8. Denoising diffusions

“

Oconnor og Ho

Ho Denoising diffusion probabilistic models

- Already in VAEs we have seen the idea of taking training data x, adding noise to obtain corrupted training data $\hat{x}$ and then training an encoder-decoder model with $\hat{x}$ as features and x as labels
- The VAE learned important aspects of the data in its latent space by 'making life difficult for ourselves'
- Denoising diffusion takes this idea to the extreme

Ho forward process

- Take a data set of images that each show a red hat
- Consider one single training image x_0
- Over T iterations, add noise to x_{i-1} to obtain x_i
- The noise is generated from a Gaussian distribution $\mathcal{N}(0, \beta_i)$
- $\beta_i > 0$ can be freely selected but is normally a sequence of increasing values
- After T iterations, the picture is 'destroyed'

Ho reverse process

- Now define a model to predict x_{i-1} taking x_i as input for all $i \in [1, T-1]$
- The model can be freely selected, only requirement is that input dimension equals output dimension
- Most frequently, the model will be a UNet
- The weights of the model will be the same across time periods and across training images
- Now train the model using gradient descent

Ho reverse process contd

- We must define the loss function. Due to the nature of the Markov chain, x_i given x_{i-1} is independent from all previous variables
- In the same way, $\hat{x}_i$ given $\hat{x}_{i+1}$ is independent from all subsequent variables since it is generated by a model that takes $\hat{x}_{i+1}$ as input
- We can therefore define the loss function as the sum of the errors for each x_i for each training picture
- Since both are Gaussians, we can approximate their difference as the Kullback Leibler KL divergence which can be calculated on closed form, and therefore we can find closed form-solutions and no longer rely on Monte Carlo simulations
- The most succesful models do not predict the value of x_i but the noise $x_i - x_{i-1}$

Ho generative model

- Once the model has been trained to convergence we have one model across time periods and training data points
- Now generate a random image in the relevant format, input to the model and run it T times
- The output of the model will be a probability distribution over 784 pixels and three color channels.
- Take the discrete value that has the highest probability - this is the produced image
- The output will be an artificial image that ressembles a red hat but does not ressemble any one training image
- As in VAE, we have captured relevant dimensions of the data by destroying and recreating images
- If we want fast sampling low T, high model expressiveness high T