CIFAR CANADIAN INSTITUTE FOR ADVANCED RESEARCH | ICRA INSTITUT CANADIEN DE RECHERCHES AVANCÉES

Université de Montréal

Deep Learning for AI

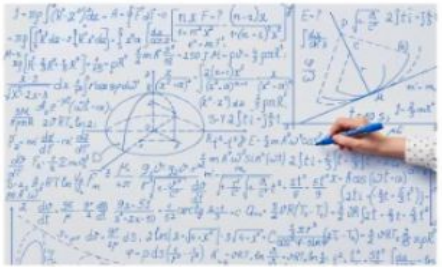
Yoshua Bengio

neuromatch academy

Mila

Neural Networks & AI: Underlying Assumption

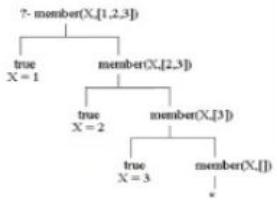
There are principles giving rise to intelligence (machine, human or animal) via learning, simple enough that they can be described compactly, i.e., our intelligence is not just the result of a huge bag of tricks and pieces of knowledge, but of general mechanisms to acquire knowledge.

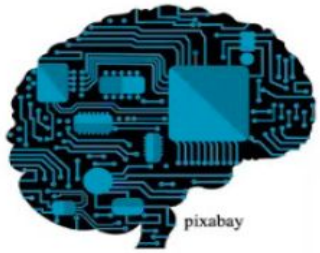


source: Marco Verch

The Machine Learning approach to AI

- **Classical AI, rule-based, symbolic**
  - knowledge is provided by humans
    - but intuitive knowledge (e.g. much of common sense) not communicable
  - machines only do inference
  - no strong learning, adaptation
  - insufficient handling of uncertainty
  - not grounded in low-level perception and action





- **Machine learning tries to fix these problems**
  - succeeded to a great extent
  - only clear route to general principles vs huge back of tricks: they can be learned.
  - higher-level (conscious) cognition not achieved yet

The Neural Net / Deep Learning Approach to AI

- **Brain-inspired**
- Synergy of a large number of simple adaptive computational units
- Focus on distributed representations
  - E.g. word representations (Bengio et al NIPS'2000), patterns of activations
- View intelligence as arising from combining
  - an objective or reward function
  - an approximate optimizer (learning rule)
  - an initial architecture / parametrization
- End-to-end learning

  (all the pieces of the puzzle adapt to help each other)

pixabay

ML 101. Learning by heart vs generalizing well

Learning by heart is easy for computers (memory, files),
difficult for humans

Generalization is not trivial and requires some assumptions

- E.g. smoothness prior

$$x \approx x' \quad \Rightarrow \quad f(x) \approx f(x')$$

Capacity: number of arbitrary examples a learner can always learn by heart

Generalization gap: difference between training and test error

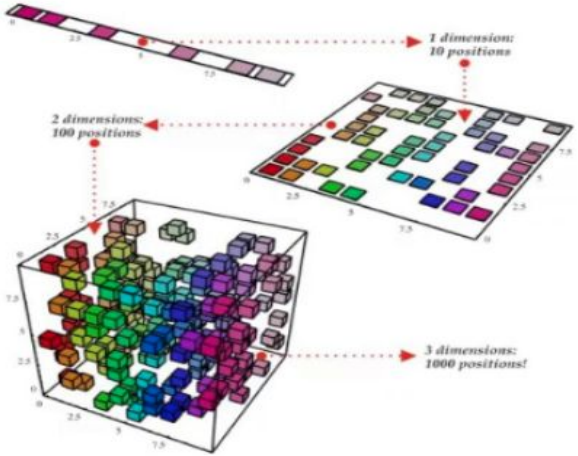
Too high capacity: overfitting (gap increases with capacity)

Too little capacity: underfitting (gap decreases with capacity)

ML 101. What We Are Fighting Against: The Curse of Dimensionality

To **generalize** locally: need representative examples for all relevant variations!

Classical solution: hope for a smooth enough target function, or make it smooth by handcrafting good features / kernel



Bypassing the curse of dimensionality

We need to build compositionality into our ML models

Just as human languages exploit compositionality to give representations and meanings to complex ideas

Exploiting compositionality: exponential gain in representational power

Distributed representations / embeddings: feature learning

Current deep architectures: multiple levels of feature learning

System 2 deep learning: compose a few concepts at a time

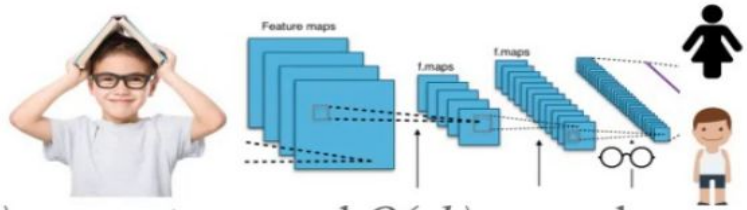
Prior assumption: compositionality useful to describe the world around us efficiently

Deep Learning: Learning an Internal Representation

- Unlike other ML methods with either
  - no intermediate representation (linear)
  - or fixed (generally very high-dimensional) intermediate representations (SVMs, kernel machines)
- What is a good representation? Makes other tasks easier.

Each feature can be discovered without the need for seeing the exponentially large number of configurations of the other features

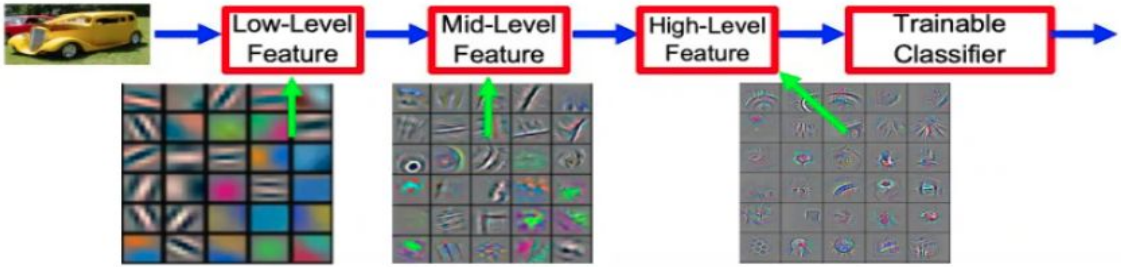
- Consider a network whose hidden units discover the following features:
  - Person wears glasses
  - Person is female
  - Person is a child
  - Etc.



- If each of n feature requires $O(k)$ parameters, need $O(nk)$ examples
- Parallel composition of features: can be exponentially advantageous
- Non-distributed non-parametric methods would require $O(n^d)$ examples

Why Multiple Layers? The World is Compositional

- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable feature transform
- Image recognition: Pixel → edge → texton → motif → part → object
- Text: Character → word → word group → clause → sentence → story
- Speech: Sample → spectral band → sound → … → phone → phoneme → word

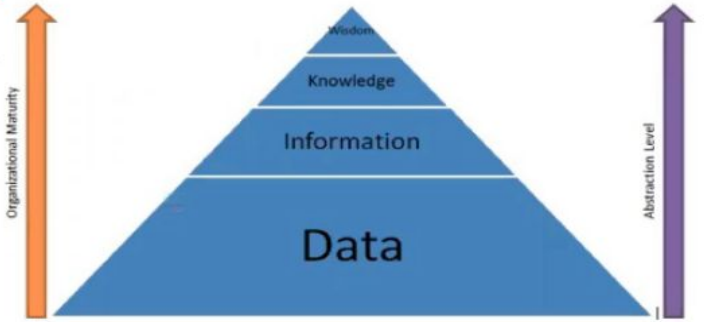


(Yann LeCun)

Learning Multiple Levels of Abstraction

- The big payoff of deep learning is to allow learning higher
  levels of abstraction
- Higher-level abstractions disentangle the factors of variation, which allows much easier generalization and transfer
- New objective: disentangle the underlying causal mechanism which explain the data at the top level

(Bengio & LeCun 2007)

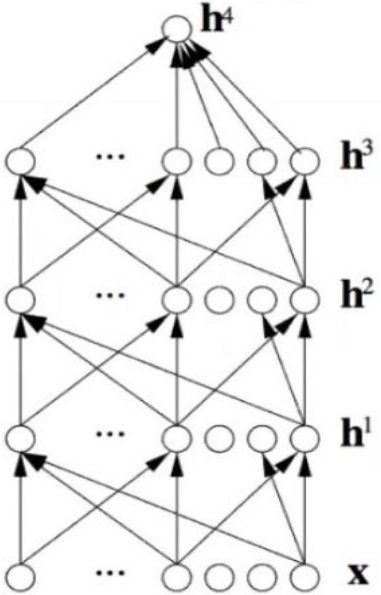


Multilayer network as universal approximator

A series of non-linear transformations of the same type but different parameters

A single but large enough hidden layer yields a universal approximator

More layers allow representing more complex functions with less parameters



Universal approximator property does not guarantee

1. easy optimization (low training error is found)
2. good generalization

Iterative training by SGD

(from Hugo Larochelle)

Topics: stochastic gradient descent (SGD)

- Algorithm that performs updates after each example
  - initialize $\boldsymbol{\theta}$ $(\boldsymbol{\theta} \equiv \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \ldots, \mathbf{W}^{(L+1)}, \mathbf{b}^{(L+1)}\})$
  - for N iterations
    - for each training example $(\mathbf{x}^{(t)}, y^{(t)})$

      $\Delta = -\nabla_{\boldsymbol{\theta}} l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)}) - \lambda \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$

      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \Delta$

      } training epoch = iteration over **all** examples
- To apply this algorithm to neural network training, we need
  - the loss function $l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - a procedure to compute the parameter gradients $\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - the regularizer $\Omega(\boldsymbol{\theta})$ (and the gradient $\nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$)
  - initialization method

Motivation for backpropagation: gradient-based optimization

- Knowing how a small change of parameters influences loss
  L tells us how to change the parameters θ

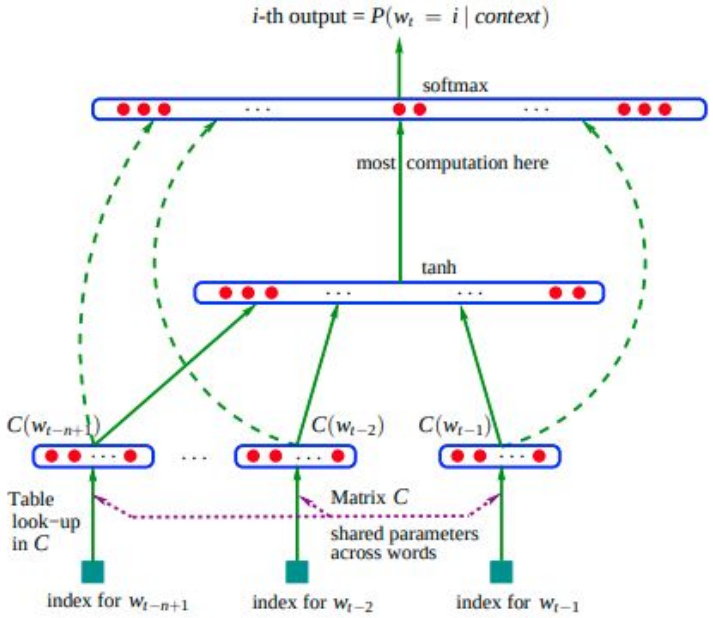
- The gradient $\frac{\partial L}{\partial \theta}$ measures the ratio of error change due to a small parameter change.

- Indicates the best local descent direction!

Neural Language Model



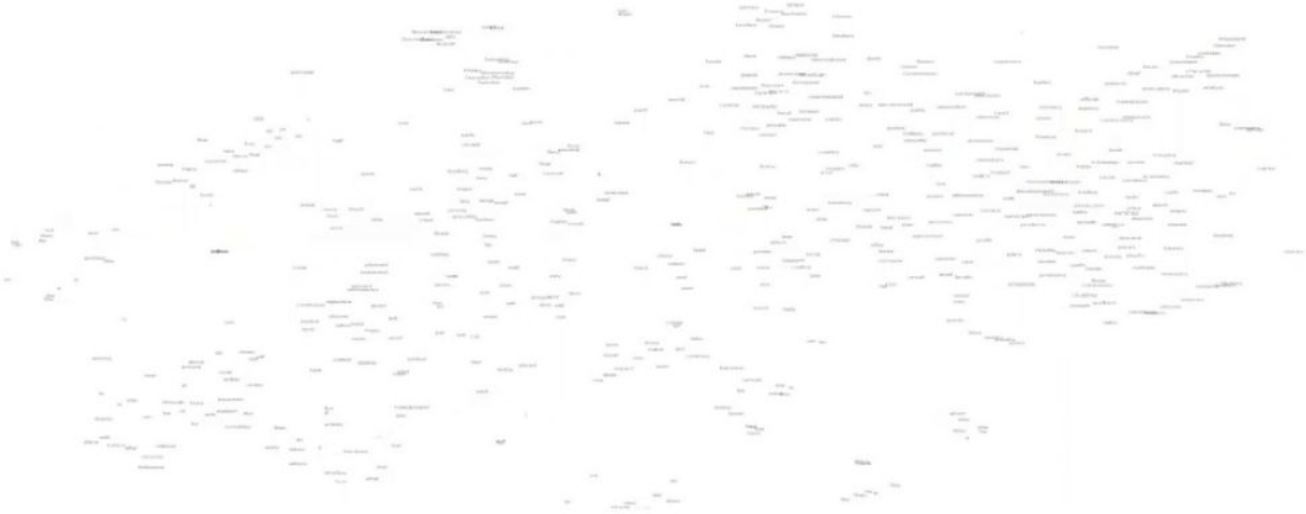
- Each word represented by a distributed continuous-valued code vector = embedding
- Generalizes to sequences of words that are semantically similar to training sequences



$$P(w_1, w_2, w_3, \ldots, w_T) = \prod_t P(w_t | w_{t-1}, w_{t-2}, \ldots, w_1)$$

(Bengio et al NIPS'2000 and JMLR 2003 "A Neural Probabilistic Language Model")

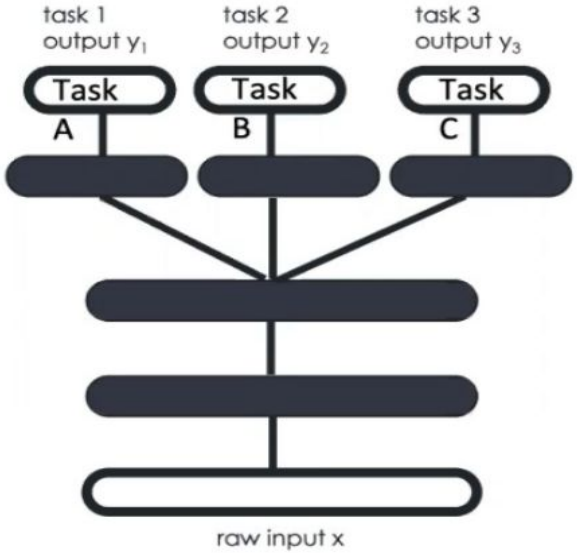
Neural word embeddings - visualization



Multi-Task Learning

- Generalizing better to new tasks (tens of thousands!) is crucial to approach AI
- Deep architectures learn good intermediate representations that can be shared across tasks

Good representations that disentangle underlying factors of variation make sense for many tasks because each task concerns a subset of the factors



Prior: shared underlying explanatory factors between tasks

(Collobert & Weston ICML 2008, Bengio et al AISTATS 2011)

E.g. dictionary, with intermediate concepts re-used across many definitions

Generative adversarial networks (GANs): a two player game with neural networks

Given:

Samples from a **target distribution** $\mathbb{P}$

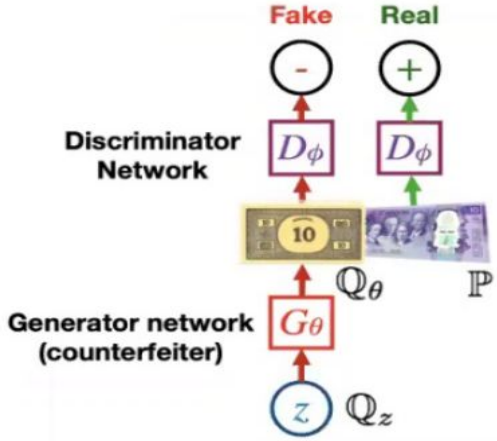
(Simple) prior $\mathbb{Q}_z$

Player 1: Generator

A neural network with parameters, θ, whose samples **fool the discriminator**

Player 2: Discriminator

Distinguish (classify) real and fake correctly



(Goodfellow et. Al & Bengio, 2014)

Generative Adversarial Networks

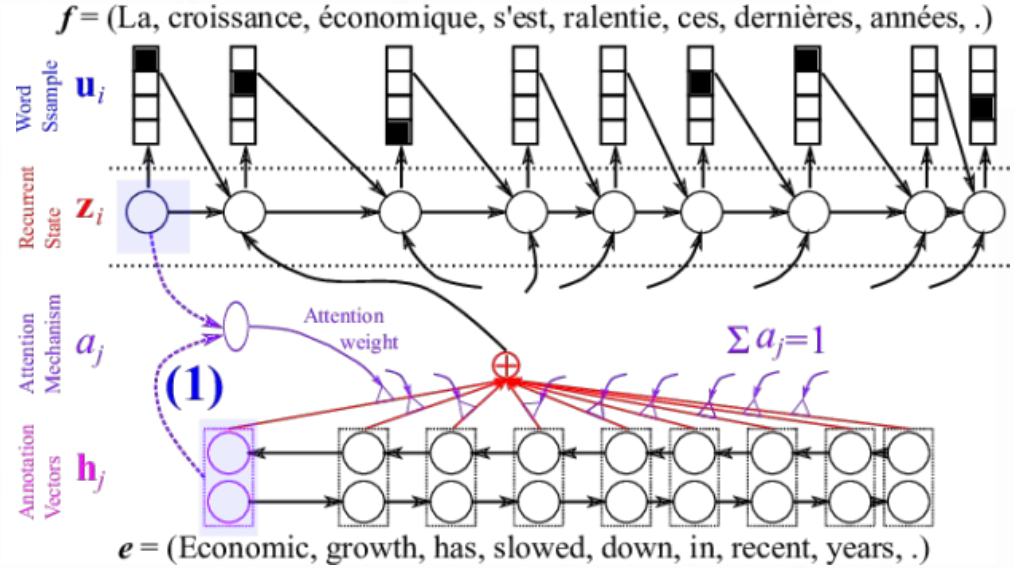
(Goodfellow et al & Bengio NIPS 2014)

(Xu et al 2018, AttnGAN)

Gating for Attention-Based Neural Machine Translation

Soft-matching of key & value:

input seen as a set of
(key, value) objects



(Bahdanau, Cho & Bengio, arXiv sept. 2014, Jean, Cho, Memisevic & Bengio, arXiv dec. 2014, Related to earlier Graves 2013 for generating handwriting)

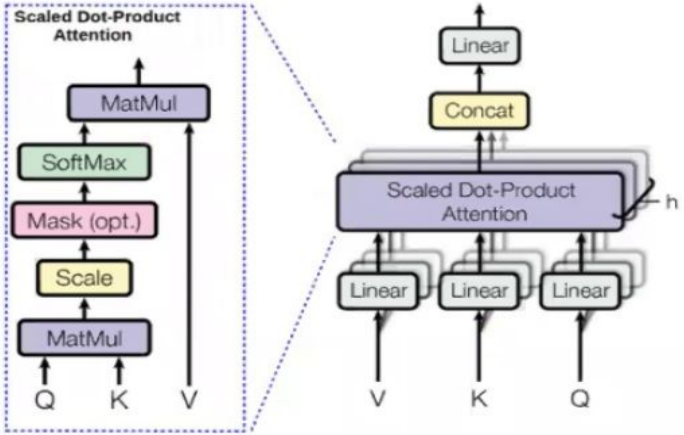
Multi-Head Attention

We can run multiple attention mechanisms in parallel to focus on different aspects of the data

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

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V).$$

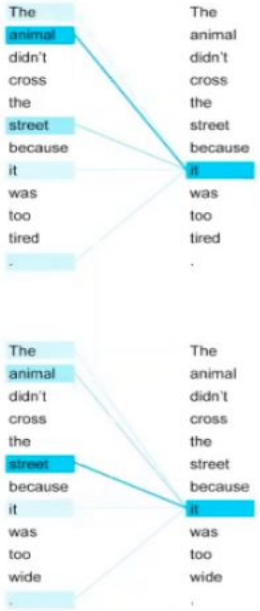
$$MultiHeadAttention(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$



(Michal Chromiak's blog)

Self-Attention & Transformers

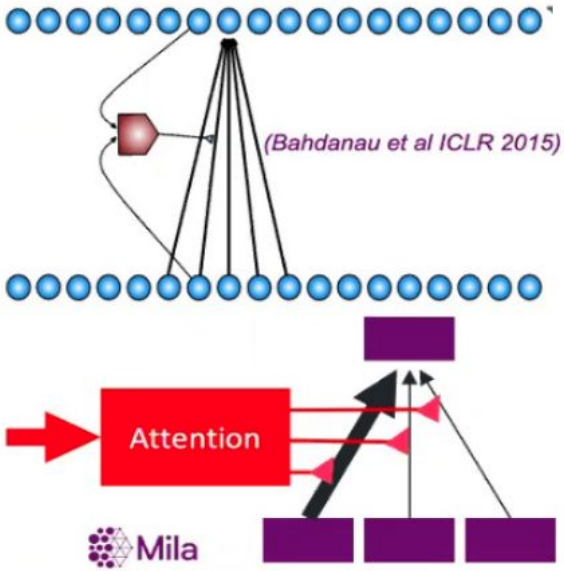
- Use attention on previous computation
- Parallelize encoder
- Encode location of each item, no need for RNN
- Transform each location based on attention from all others
- See also *Sparse Attentive Backtracking*, Ke et al NIPS'2018



(Lin et al ICLR 2017; Vaswani et al NIPS'2017)

Core Ingredient For Conscious Processing: **Attention**

- Focus on a one or a few elements at a time
- Content-based soft attention is convenient, can backprop to *learn where to attend*
- Attention is an internal action, needs a learned attention policy
- Operating on unordered SETS of (key, value) pairs
- SOTA in NLP



Missing from Current ML: Understanding & Generalization Beyond the Training Distribution

- Learning theory only deals with generalization within the same distribution
- Models learn but do not generalize well (or have high sample complexity when adapting) to modified distributions, non-stationarities, etc.
- Poor reuse, poor modularization of knowledge

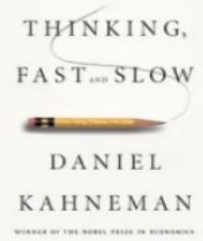
System 1 vs. System 2 Cognition

2 systems (and categories of cognitive tasks):

Manipulates high-level / semantic concepts, which can be recombined combinatorially

System 1

- Intuitive, fast, **UNCONSCIOUS**, 1-step parallel, non-linguistic, habitual
- Implicit knowledge
- Current DL



System 2

- Slow, logical, **sequential**, **CONSCIOUS**, linguistic, algorithmic, planning, reasoning
- Explicit knowledge
- DL 2.0

Mila

ML Experiments

- Get your hands dirty and become expert at running and coding experiments
- Don't trust your own code, so try to run other people's code as baseline, make your code available to others to replicate your experiments
- Run comparisons, against both simple baseline methods and against the state-of-the-art
- When testing a new method, make sure to use at least some tasks which already are in the literature, with published results (and ideally published code), so you can compare yourself fairly (by opposition to only working on your own dataset)
- Don't trust negative experimental results: could be a bug, so find an explanation for them
- Run experiments not just to try to beat some benchmark but to answer some scientific question, to figure out why something work, to discard alternative explanatory hypotheses
- Knowledge acquisition: what can an experiment teach us we did not know or were not sure of ? How to strengthen conclusions (ablations, comparisons)? What does it work? Which ingredients matter?