Chapter 7, Part 1: Introduction to Deep Learning

Advanced Topics in Statistical Machine Learning

Tom Rainforth

Hilary 2024

rainforth@stats.ox.ac.uk

https://youtu.be/cQ54GDm1eL0

Motivation: High Dimensional Inputs

Many common data types are (very) high dimensional, e.g. text, images, and videos

Disentangling Disentanglement in Variational Autoencoders

Emile Mathieu [*1] **Tom Rainforth** [*1] **N. Siddharth** [*2] **Yee Whye Teh** [1]

Abstract

We develop a generalisation of disentanglement in variational autoencoders (VAEs)—*decomposition* of the latent representation—characterising it as the fulfilment of two factors: a) the latent encodings of the data having an appropriate level of overlap, and b) the aggregate encoding of the data conforming to a desired structure, represented through the prior. Decomposition permits disentanglement, i.e. explicit independence between latents, as a special case, but also allows for a much richer class of properties to be imposed on the learnt representation, such as sparsity, clustering, independent subspaces, or even intricate hierarchical dependency relationships. We show that the β-VAE varies from the standard VAE predominantly in its control of latent overlap and that for the standard choice of an isotropic Gaussian prior, its objective is invariant to rotations of the latent representation. Viewed from the decomposition perspective, breaking this invariance with simple manipulations of the prior can yield better disentanglement with little or no detriment to reconstructions. We further demonstrate how other choices of prior can assist in producing different decompositions and introduce an alternative training objective that allows the control of both decomposition factors in a principled manner.

1. Introduction

An oft-stated motivation for learning disentangled representations of data with deep generative models is a desire to achieve interpretability (Bengio et al., 2013; Chen et al., 2017)—particularly the *decomposability* (see §3.2.1 in Lipton, 2016) of latent representations to admit intuitive explanations. Most work has focused on capturing purely *independent* factors of variation (Alemi et al., 2017; Ansari and Soh, 2019; Burgess et al., 2018; Chen et al., 2018; 2017; Eastwood and Williams, 2018; Esmaeili et al., 2019; Higgins et al., 2016; Kim and Mnih, 2018; Xu and Durrett, 2018; Zhao et al., 2017), typically evaluating this using purpose-built, synthetic data (Eastwood and Williams, 2018; Higgins et al., 2016; Kim and Mnih, 2018), whose generative factors are independent by construction.

This conventional view of disentanglement, as recovering independence, has subsequently motivated the development of formal evaluation metrics for independence (Eastwood and Williams, 2018; Kim and Mnih, 2018), which in turn has driven the development of objectives that target these metrics, often by employing regularisers explicitly encouraging independence in the representations (Eastwood and Williams, 2018; Esmaeili et al., 2019; Kim and Mnih, 2018).

We argue that such an approach is not generalisable, and potentially even harmful, to learning interpretable representations for more complicated problems, where such simplistic representations cannot accurately mimic the generation of high dimensional data from low dimensional latent spaces, and more richly structured dependencies are required.

We posit a generalisation of disentanglement in VAEs—*decomposing* their latent representations—that can help avoid such pitfalls. We characterise decomposition in VAEs as the fulfilment of two factors: a) the latent encodings of data having an appropriate level of overlap, and b) the aggregate encoding of data conforming to a desired structure, represented through the prior. We emphasize that neither of these factors is sufficient in isolation: without an appropriate level of overlap, encodings can degrade to a lookup table where the latents convey little information about data, and without the aggregate encoding of data following a desired structure, the encodings do not decompose as desired.

Disentanglement *implicitly* makes a choice of decomposition: that the latent features are independent of one another. We make this *explicit* and exploit it to both provide improvement to disentanglement through judicious choices of structure in the prior, and to introduce a more general framework flexible enough to capture alternate, more complex, notions of decomposition such as sparsity, clustering, hierarchical structuring, or independent subspaces.

[*]Equal contribution [1]Department of Statistics [2]Department of Engineering, University of Oxford. Correspondence to: Emile Mathieu <emile.mathieu@stats.ox.ac.uk>, Tom Rainforth <rainforth@stats.ox.ac.uk>, N. Siddharth <nsid@robots.ox.ac.uk>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

Motivation: Flexibility and Complicated Output Spaces

Figure 1: Fake Rembrandt generated by a deep learning algorithm.
Image credit: `https://www.nextrembrandt.com/`

Motivation: Performing Well on Huge Datasets

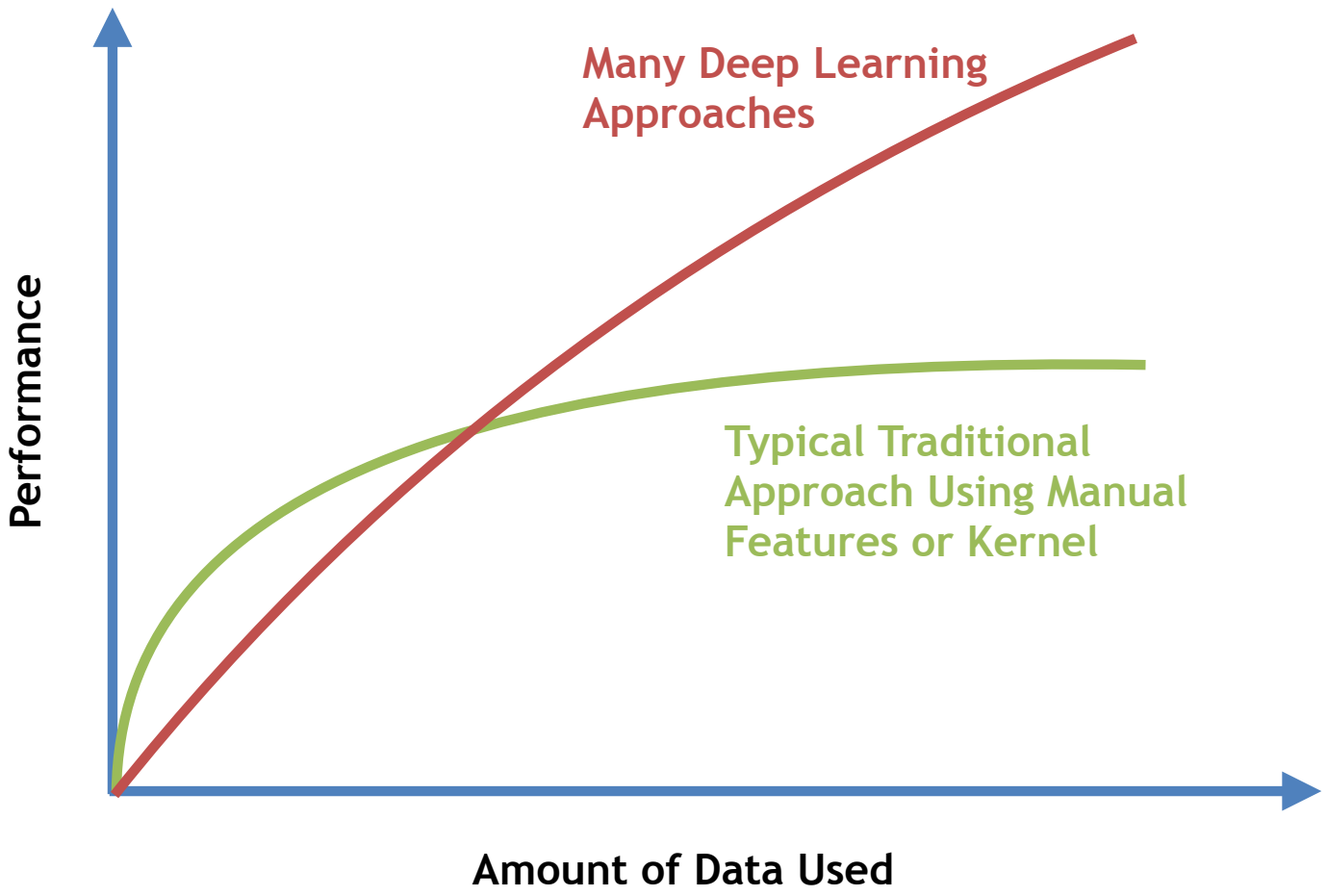

Figure 2: Characterization of deep learning method's performance as training dataset size increases (somewhat overgeneralized)

Motivation: Graceful Computational Scaling

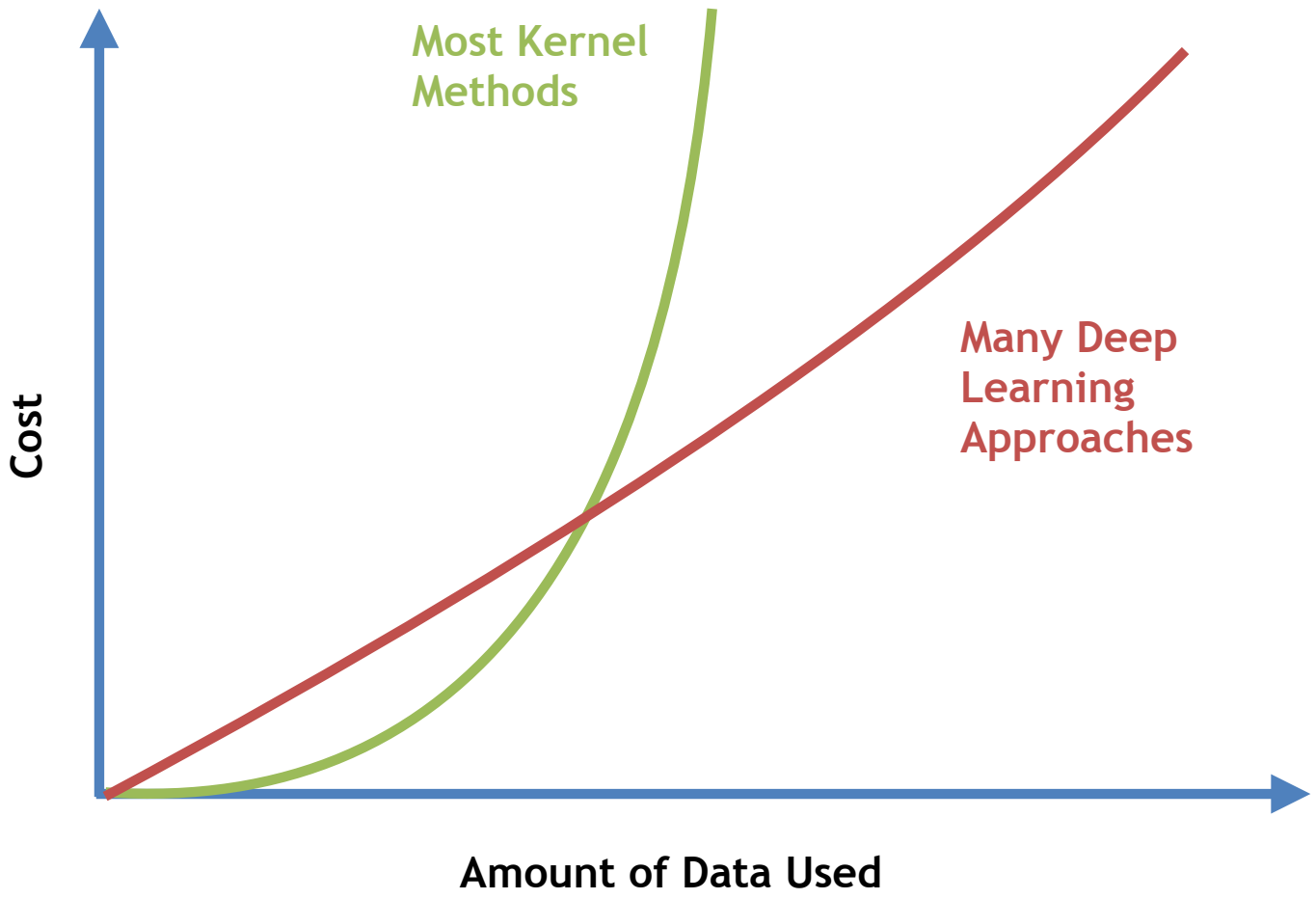

Figure 3: Characterization of deep learning method's cost as training dataset size increases (somewhat overgeneralized)

Feature Engineering

Even though they imply an infinite number of features, kernel methods still require an effective similarity function between inputs and that requires specialist knowledge about the space of inputs

- To know if inputs are similar, we need to know what about them is important

We can think of this problem as that of **feature engineering**: to perform effective machine learning we need effective **representations** of our inputs that encapsulate salient information

- We need to pick out **patterns** in the space of inputs that allow us to assess similarities and differences between datapoints
- We want to **ignore** unimportant aspects and noise

Manually constructing effective features (or kernels) is extremely difficult in **high dimensions**

End–To–End Learning

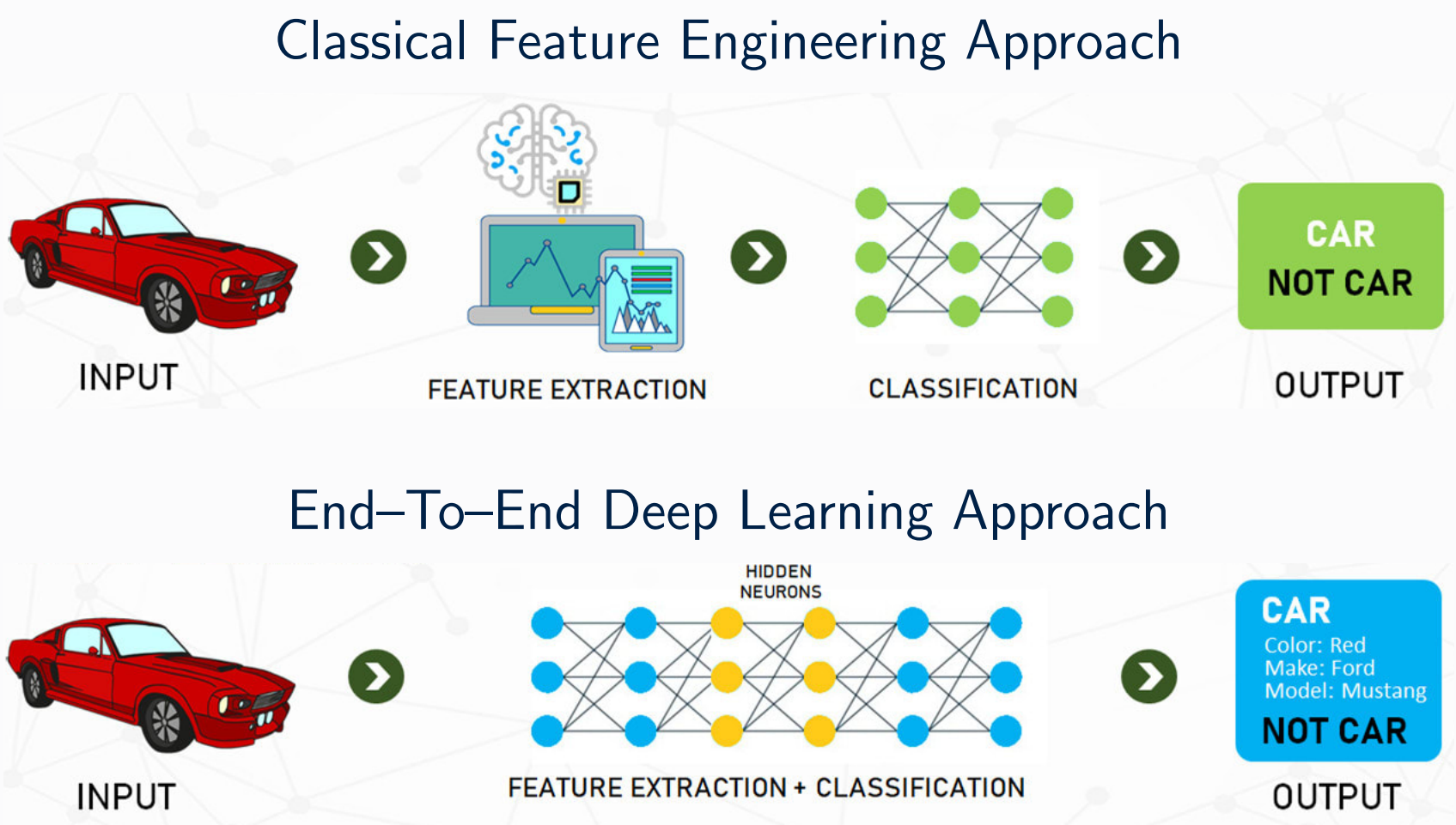

Figure 4: Deep learning looks to replace manual feature engineering with an end–to–end approach that learns everything at the same time

End–To–End Learning (2)

A classical feature–engineering approach will take a fixed feature mapping $\varphi : \mathcal{X} \rightarrow \mathbb{R}^m$ and then perform ERM to optimize the parameters of a predictive model $f_\theta : \mathbb{R}^m \rightarrow \mathcal{Y}$

$$\theta^* = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\theta(\varphi(x_i))) + r(\theta)$$

where L is our loss function and r is a regularize.

The starting point for deep learning is to instead take an end–to–end approach where we also parameterize the feature mapping $\varphi_\eta : \mathcal{X} \rightarrow \mathbb{R}^m$ and then optimize all parameters

$$\{\theta^*, \eta^*\} = \arg\min_{\theta,\eta} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\theta(\varphi_\eta(x_i))) + r(\theta, \eta)$$

We are thus replacing manual feature engineering with **feature learning** that is performed simultaneously to training the model

Using Multiple Layers

The key idea of deep learning is now to take this a step further and introduce hierarchical **composition** of **nonlinear** functions

We do this by replacing $f_\theta(\varphi_\eta(x))$ with

$$f^m_{\theta^m}\left(f^{m-1}_{\theta^{m-1}}\left(\dots f^2_{\theta^2}\left(f^1_{\theta^1}\left(x\right)\right)\dots\right)\right)$$

such that our overall predictive model is $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$ is given by

$$f_\theta = f^m_{\theta^m} \circ f^{m-1}_{\theta^{m-1}} \circ \dots \circ f^2_{\theta^2} \circ f^1_{\theta^1}$$

We can also think of this as having multiple **layers** of features, $h^\ell \in \mathbb{R}^{d_\ell}$, such that $h^0 = x$, $h^m = f_\theta(x)$ and

$$h^\ell = f^\ell_{\theta^\ell}\left(h^{\ell-1}\right)$$

"**Deep**" now refers to having many such layers, where we **learn** the parameters θ^ℓ for each of the associated mappings

Deeper Layers Represent More Complex Features

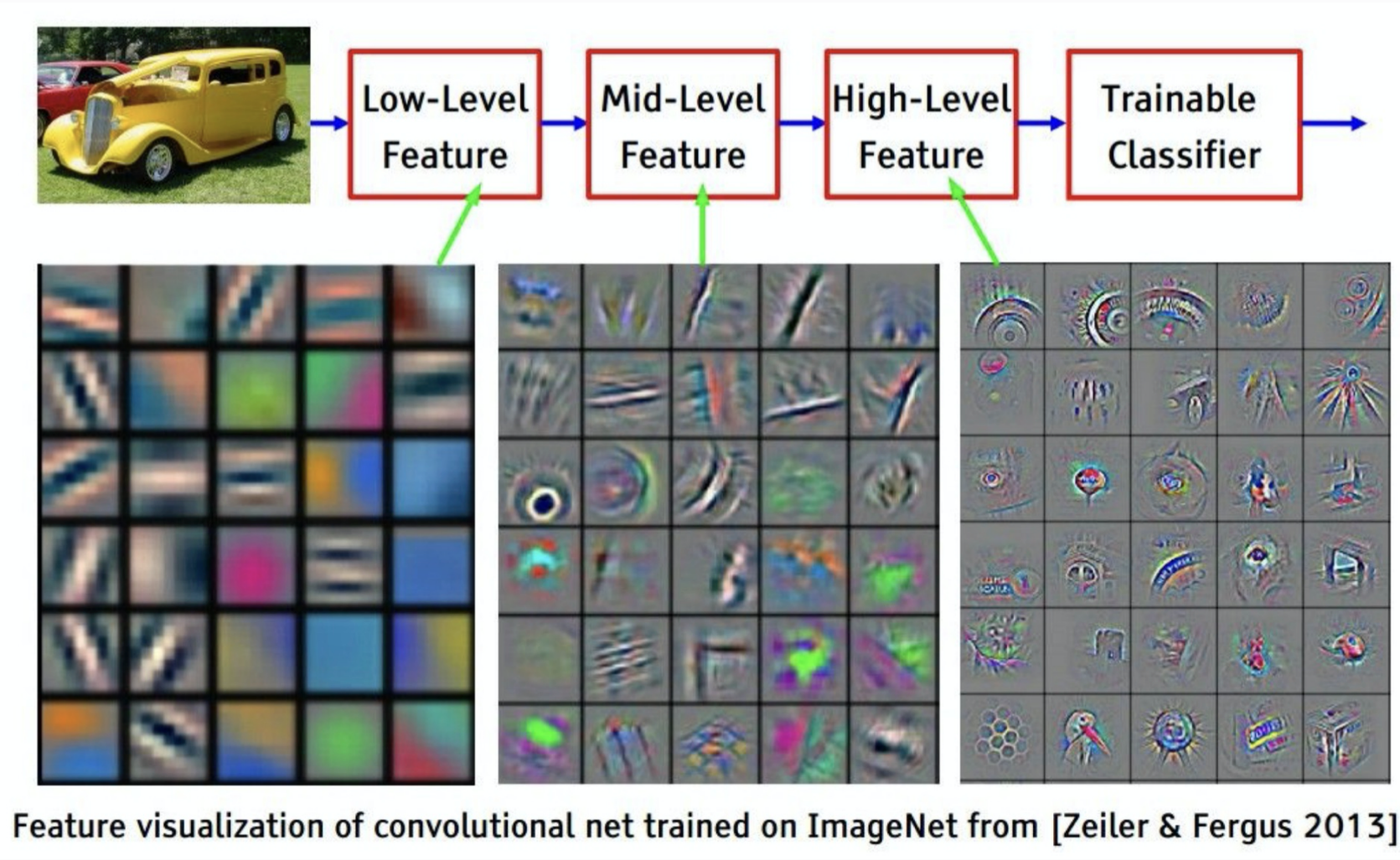

Figure 5: Visualization of features represented by different layers. Individual images in the grid correspond to example dimensions of a h^{ℓ}.

Desiderata for f_θ

To more precisely understand why taking a deep approach is beneficial, let's first consider the desirable properties for the class of predictors f_θ:

- **Powerful**: we want f_θ to be able to accurately capture complex input–output relationships
- **Flexible and Expressive**: we want to be able to apply the approach to a wide array of problems and use customized setups that are tailored to specific tasks
- **Easily Trainable and Scalable**: we need to be able to effectively and tractably train good predictors, even with large amounts of high dimensional training data.

Achieving these Desiderata

- To be sufficiently powerful with large quantities of data, we need to be able to use a **large number of parameters** θ
- To allow effective training in high dimensions we will typically need to be able to take **derivatives** $\nabla_\theta f_\theta(x)$
- To be flexible and expressive, we will generally require complex hypothesis classes to be constructible by piecing together simpler building blocks

Deep learning is a compositional framework that achieves these by building up complex hypothesis classes from compositions of simple **differentiable** building blocks

A particular configuration of these building blocks represents a hypothesis class and is known as an **architecture**

Deep Learning

Deep learning is a machine learning approach where we construct a parameterized predictive function using a computational graph of simpler differentiable functional blocks, and then train this on data using some form of gradient-based optimization.

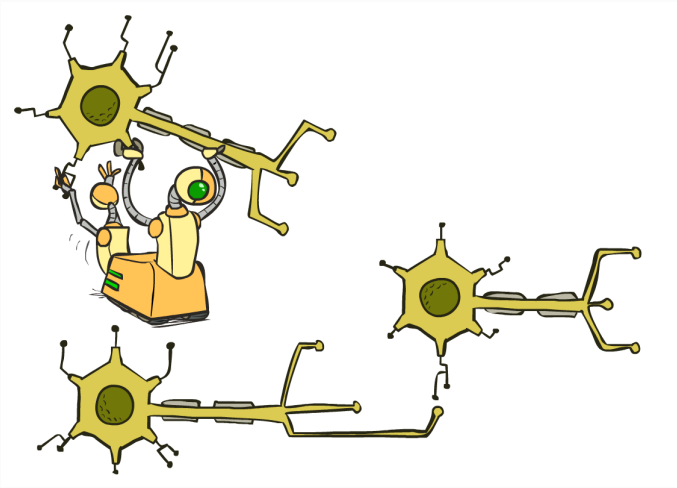

Image credit: Dan Klein and Pieter Abbeel

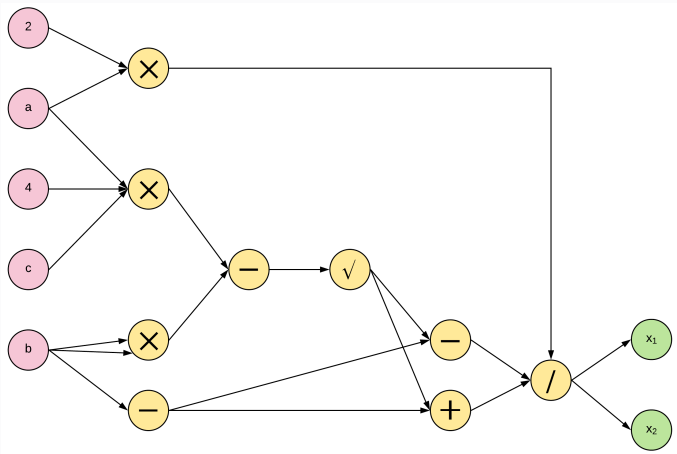

Image credit: Ekaba Bisong

Differentiable Programing

Yann LeCun

January 5 at 10:13pm ·

OK, Deep Learning has outlived its usefulness as a buzz-phrase.
Deep Learning est mort. Vive Differentiable Programming!

Yeah, Differentiable Programming is little more than a rebranding of the modern collection Deep Learning techniques, the same way Deep Learning was a rebranding of the modern incarnations of neural nets with more than two layers.

But the important point is that people are now building a new kind of software by assembling networks of parameterized functional blocks and by training them from examples using some form of gradient-based optimization.

Compositions of Differentiable Functions are Differentiable

The reason we can do this is because the chain rule ensures that the composition of differentiable functions is itself differentiable:

$$\frac{\partial L(y_i, f_\theta(x_i))}{\partial \theta^k} = \frac{\partial L(y_i, f_\theta(x_i))}{\partial h^m} \frac{\partial h^m}{\partial h^{m-1}} \cdots \frac{\partial h^{k+1}}{\partial h^k} \frac{\partial h^k}{\partial \theta^k}\bigg|_{x=x_i}$$

where each $\frac{\partial h^\ell}{\partial h^{\ell-1}}$ is a $d_\ell \times d_{\ell-1}$ matrix representing the Jacobian of $f^\ell_{\theta^\ell}$ with respect to its inputs.

Note that, except for the first layer, we require that the $f^\ell_{\theta^\ell}$ are differentiable with respect to their parameters **and** their inputs

This also holds for general differentiable **computation graphs** by allowing individual h^ℓ to be split up into separate nodes $h^\ell = \left[h^{\ell,1}, h^{\ell,2}, \ldots, h^{\ell,t_\ell}\right]$, or equivalently grouping nodes together as layers and padding with identity mappings as required

Computation Graphs

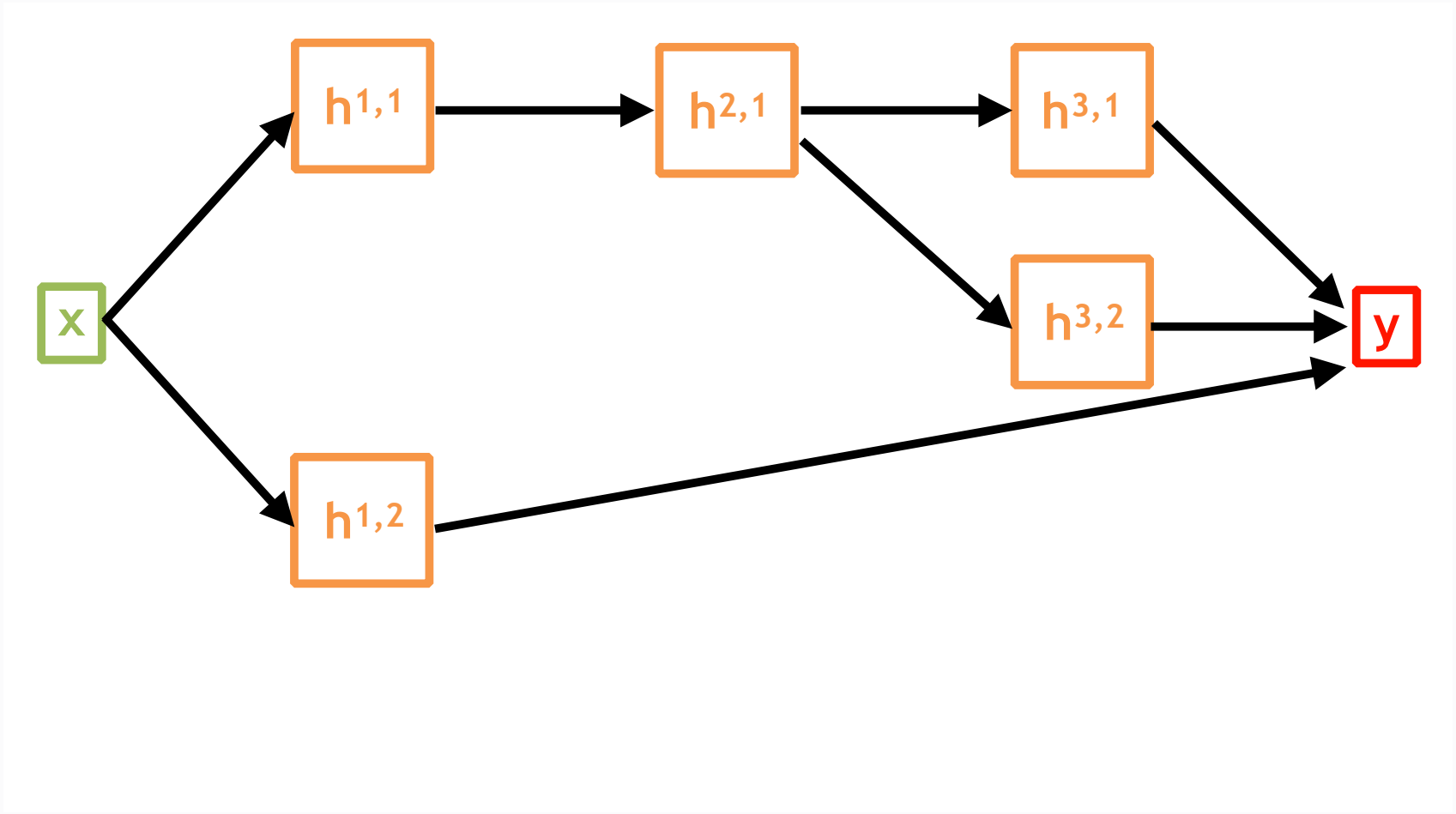

Figure 6: Example computation graph and how this can be represented as series of layers by grouping and padding nodes. id represents the identity function $f(x) = x$.

Computation Graphs

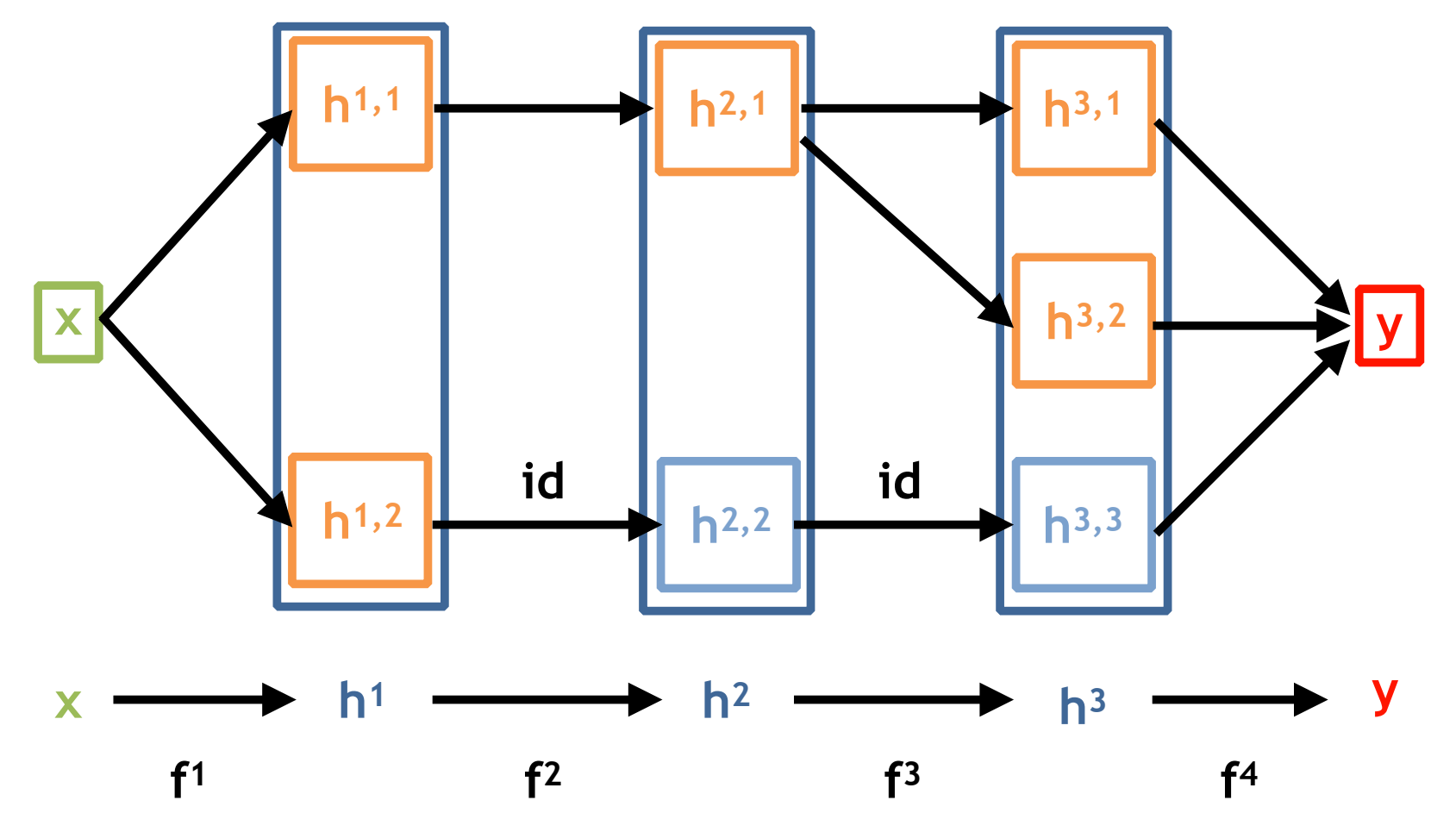

Figure 6: Example computation graph and how this can be represented as series of layers by grouping and padding nodes. id represents the identity function $f(x) = x$.

Applications: Image Classification

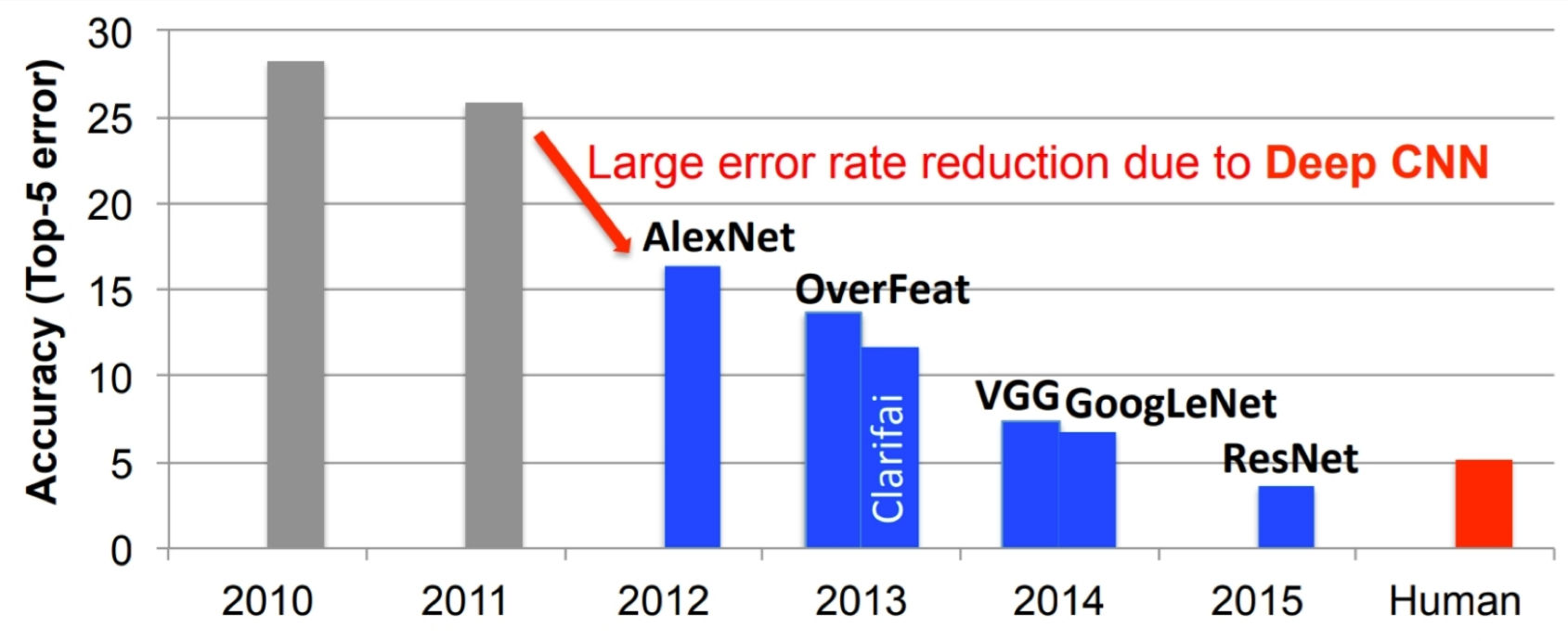

Figure 7: State of the art performance on ImageNet (a large image classification dataset) over time. Figure taken from Russakovsky et al IJCV 2015.

Applications: Image Captioning

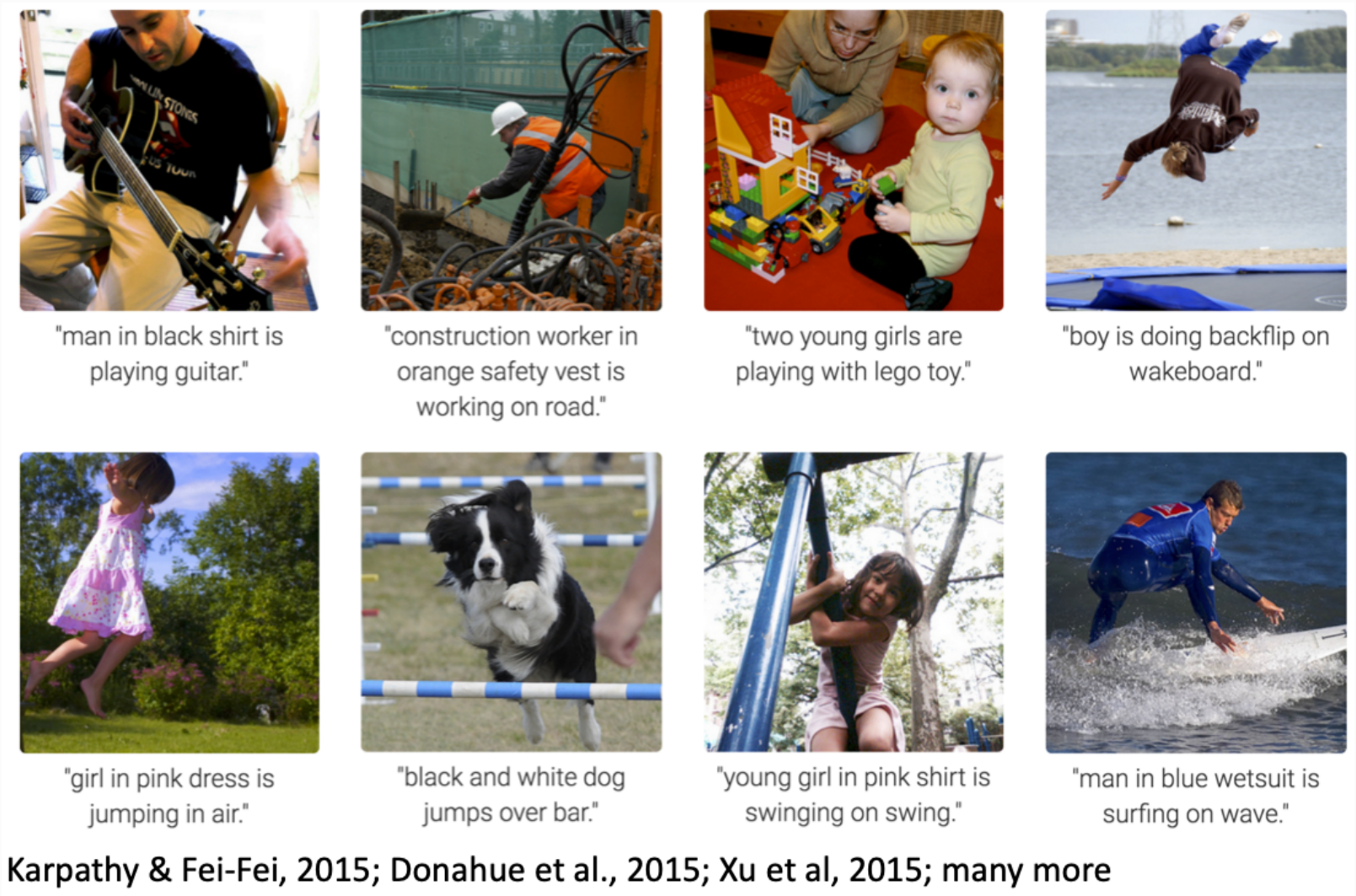

Figure 8: MS COCO Image captioning challenge. Image credit Dan Klein and Pieter Abbeel

Applications: Deep Fakes

https://youtu.be/h_jrebvmPlk

Applications: Test Translation

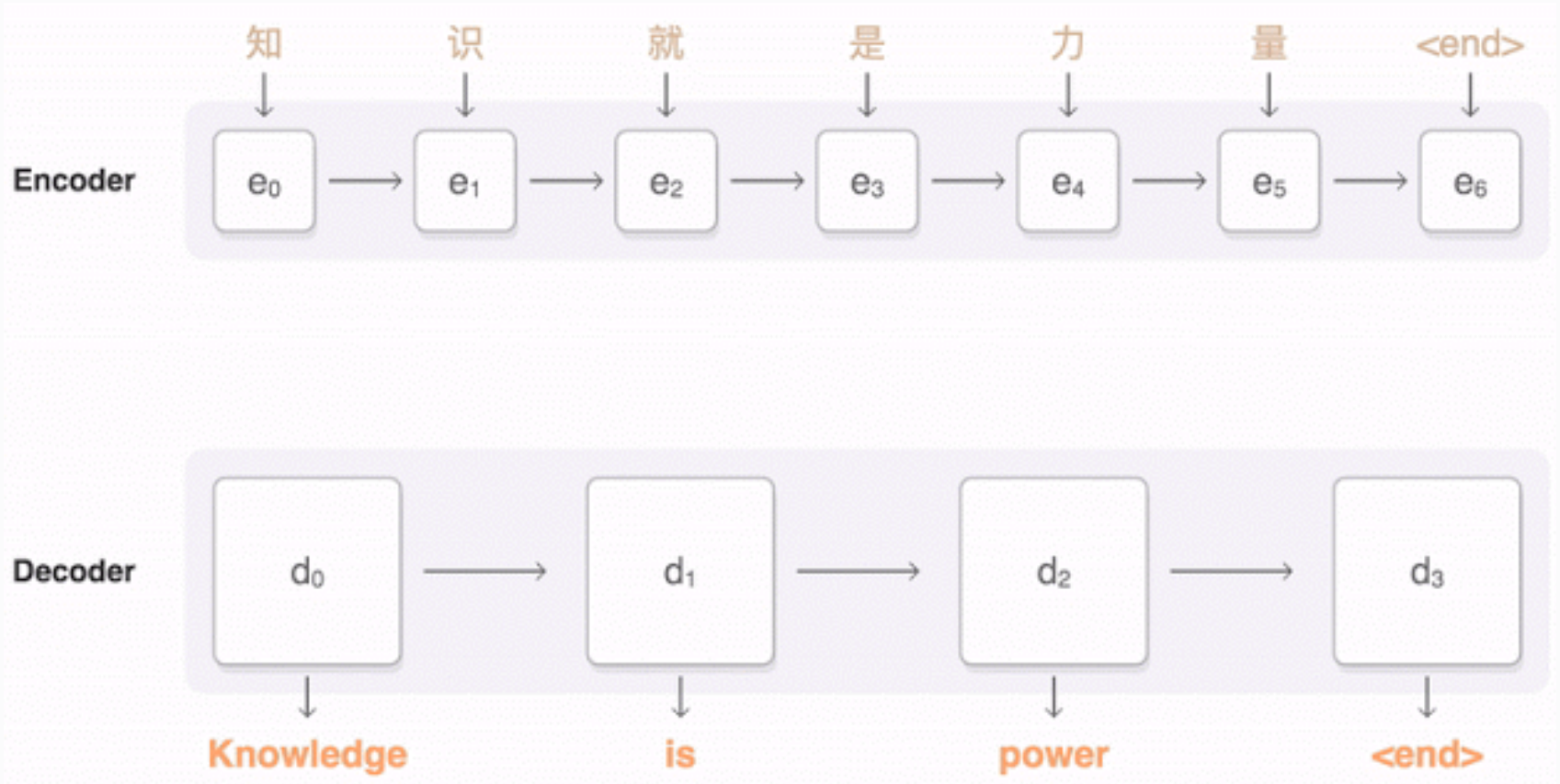

Figure 9: Translating Cantonese into English. Image Credit: Google AI Research Blog

Applications: Large Language Models

https://chat.openai.com/chat

Applications: Learning to Play Games

Learning to play Mario Kart
`https://www.youtube.com/watch?v=Tnu4O_xEmVk`

Beating top players at Starcraft
`https://youtu.be/DMXvkbAtHNY`

Learning to park (shows requirement for a lot of training/simulated data)
`https://youtu.be/VMp6pq6_QjI`

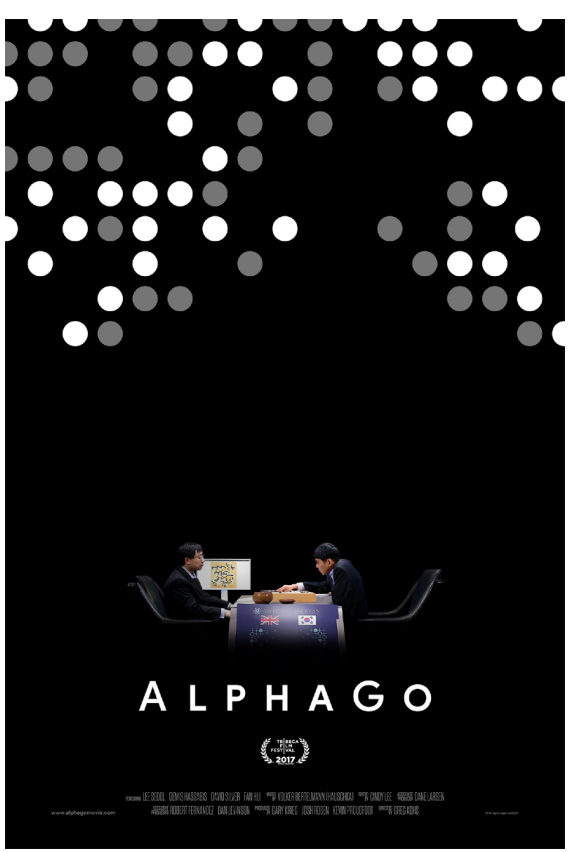

Figure 10: Beating the world champion Go player