



CENG 403

Introduction to Deep Learning

Week 12a

Sinan Kalkan

Class Activation Maps

- Weighted combination of the feature maps before GAP:

$$M(x, y) = \sum_k w_k^c f_k(x, y)$$

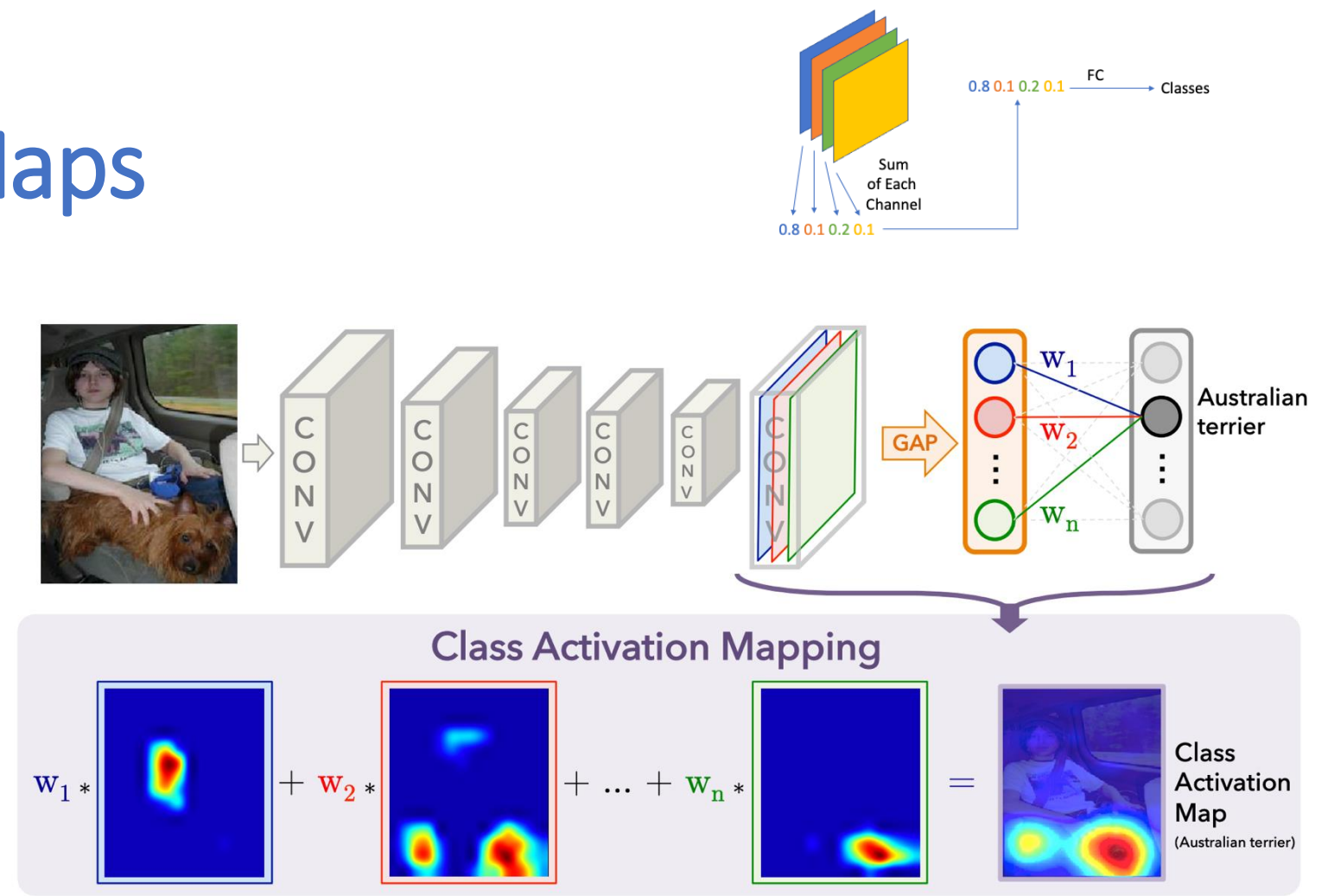


Figure 2. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2921–2929.

Class Activation Maps

- GradCAM:

$$\alpha_k^c = \sum_{x,y} \frac{\partial S_c}{\partial f_k(x,y)}$$

$$M^c(x,y) = \text{ReLU} \left(\sum_k \alpha_k^c f_k(x,y) \right)$$


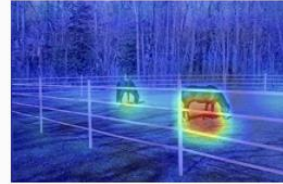

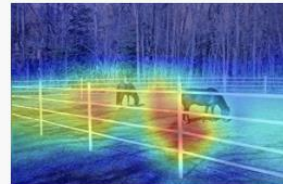
Network	Image	GradCAM
VGG16		
Resnet50		

Figure: <https://pypi.org/project/grad-cam/>

R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization," arXiv preprint arXiv:1610.02391, 2016.

Chattopadhyay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018, March). Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 839-847). IEEE.

Feature inversion with perceptual losses

- Reconstruct an image from its representation

This section introduces our method to compute an approximate inverse of an image representation. This is formulated as the problem of finding an image whose representation best matches the one given [34]. Formally, given a representation function $\Phi : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^d$ and a representation $\Phi_0 = \Phi(\mathbf{x}_0)$ to be inverted, reconstruction finds the image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ that minimizes the objective:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x}) \quad (1)$$

where the loss ℓ compares the image representation $\Phi(\mathbf{x})$ to the target one Φ_0 and $\mathcal{R} : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}$ is a regulariser capturing a *natural image prior*.

Understanding Deep Image Representations by Inverting Them

Aravindh Mahendran
University of Oxford

Andrea Vedaldi
University of Oxford

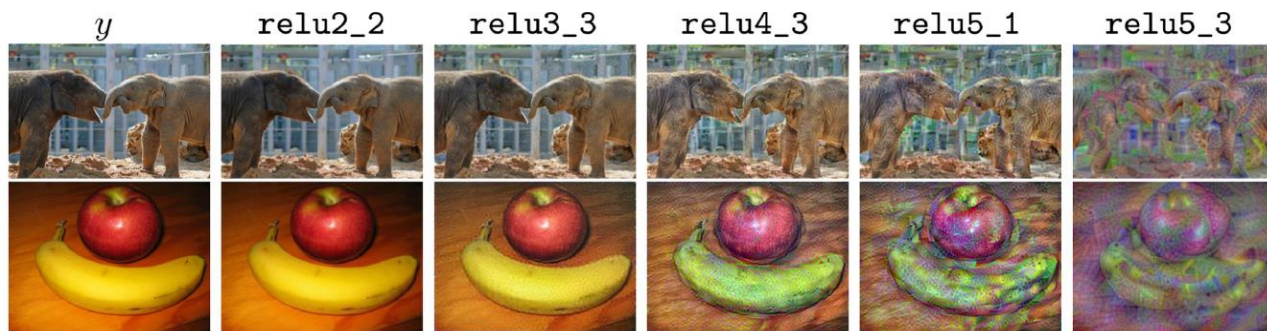


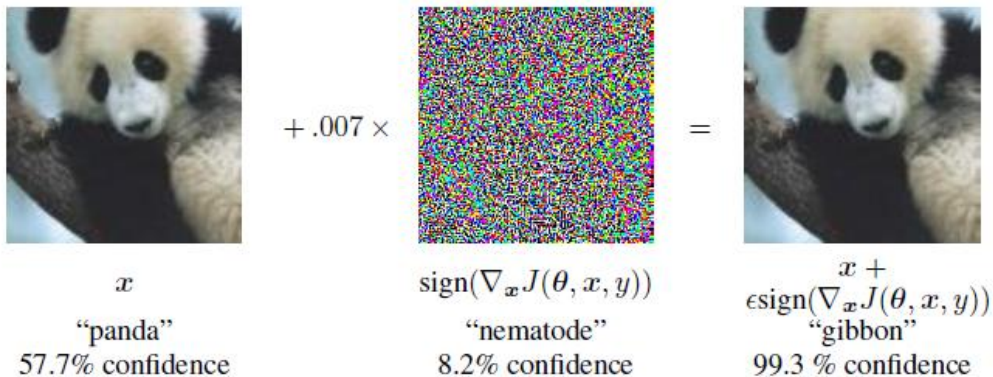
Figure from Johnson, Alahi, and Fei-Fei, “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”, ECCV 2016.

Fooling ConvNets

Previously on CENG403

- Given an image I labeled as y_1 , find minimum “ r ” (noise) such that $I + r$ is classified as a different label, y_2 .
- I.e., minimize:

$$\arg \min_r \text{loss}(I + r, y_2) + c|r|$$



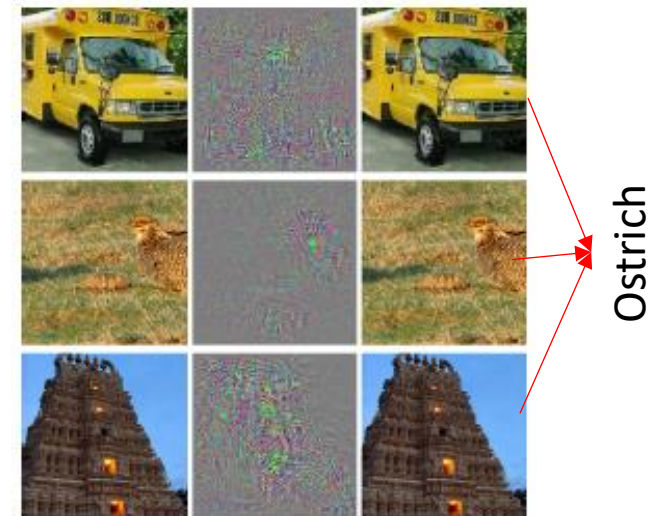
Sinan Kalkan

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES

Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy
 Google Inc., Mountain View, CA
 {goodfellow, shlens, szegedy}@google.com

Intriguing properties of neural networks

Christian Szegedy Google Inc.	Wojciech Zaremba New York University	Ilya Sutskever Google Inc.	Joan Bruna New York University
Dumitru Erhan Google Inc.	Ian Goodfellow University of Montreal	Rob Fergus New York University Facebook Inc.	



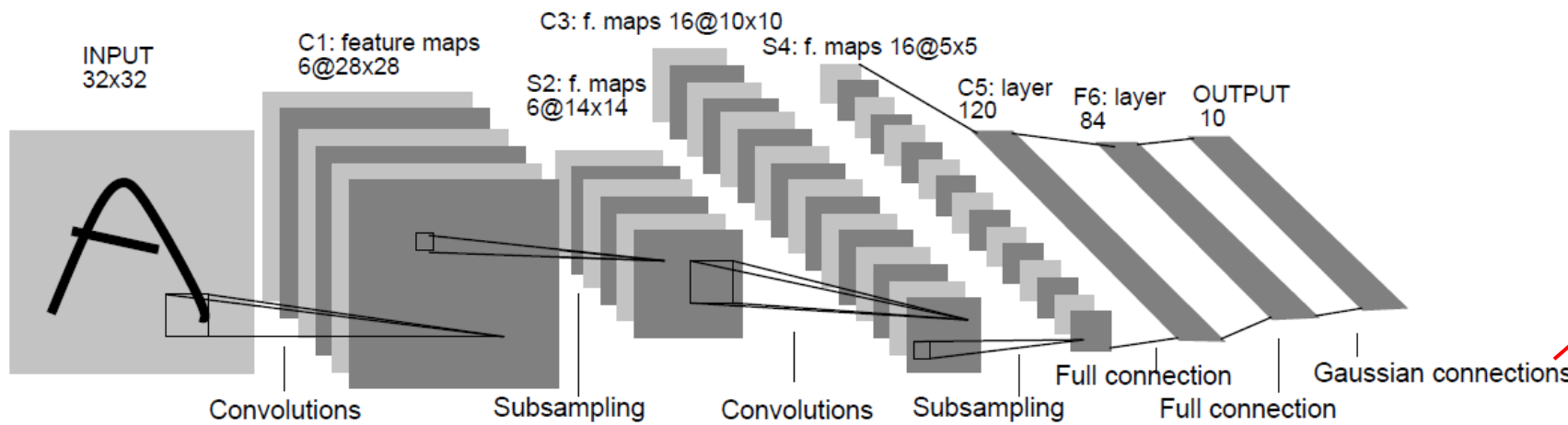
Previously on CENG403

LeNet (1998)

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

- For reading zip codes and digits



Euclidean RBF:

$$y_i = \sum_j (x_j - w_{ij})^2.$$

F=5x5 S=1 P=0	F=2x2 S=2 sigm($\alpha \times avg + b$) α & b: trainable	F=5x5 S=1 P=0	F=2x2 S=2 sigm($\alpha \times avg + b$) α & b: trainable	F=5x5 S=1 P=0
---------------------	--	---------------------	--	---------------------

AlexNet (2012)

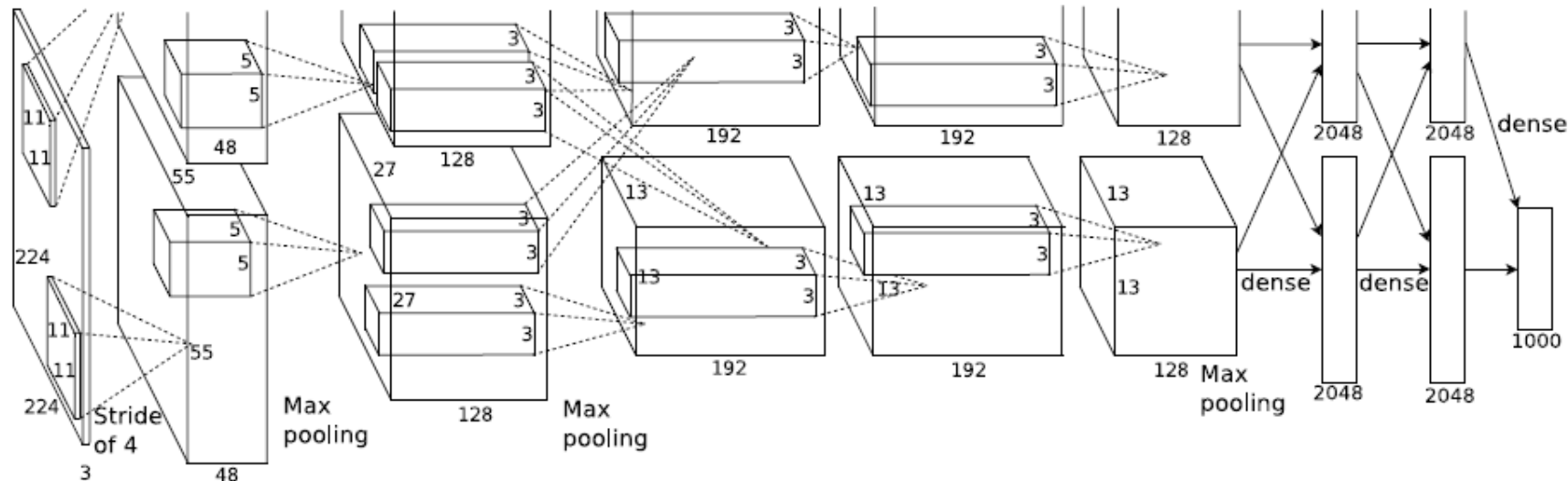
ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

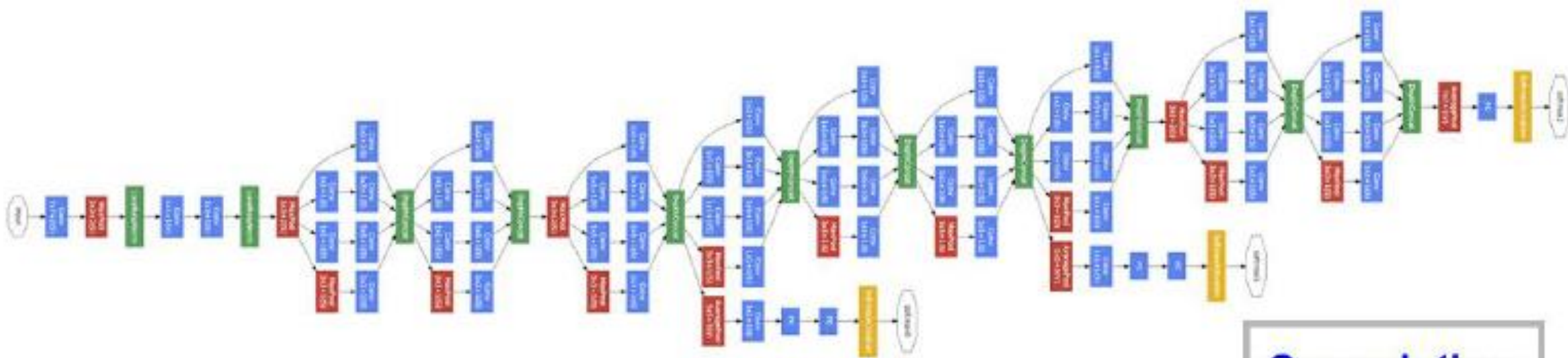
Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

- Popularized CNN in computer vision & pattern recognition
- ImageNet ILSVRC challenge 2012 winner
- Similar to LeNet
 - Deeper & bigger
 - Many CONV layers on top of each other (rather than adding immediately a pooling layer after a CONV layer)
 - Uses GPU
- 650K neurons. 60M parameters. Trained on 2 GPUs for a week.



GoogleNet (2014)

Previously on CENG403



Convolution

Pooling

Softmax

Other

Going deeper with convolutions

Christian Szegedy
Google Inc.

Wei Liu
University of North Carolina, Chapel Hill

Yangqing Jia
Google Inc.

Pierre Sermanet
Google Inc.

Scott Reed
University of Michigan

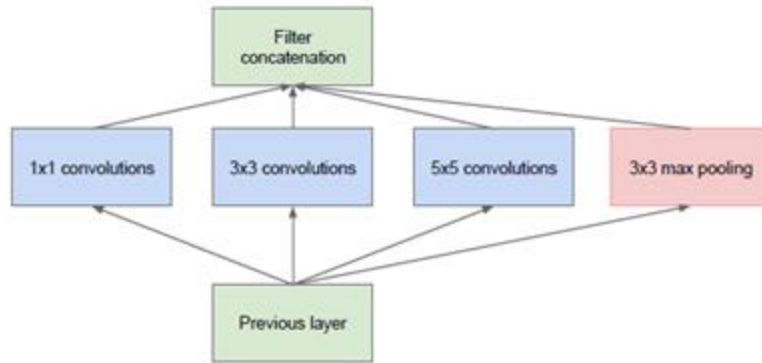
Dragomir Anguelov
Google Inc.

Dumitru Erhan
Google Inc.

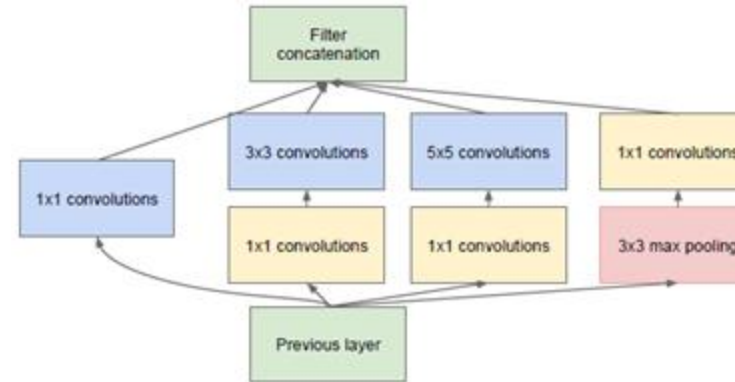
Vincent Vanhoucke
Google Inc.

Andrew Rabinovich
Google Inc.

Inception module: “network in network” (inspired from Lin et al., 2013)



(a) Inception module, naïve version



(b) Inception module with dimension reductions

- Concatenation is performed along the “channels” (depth).
 - The output of inception layers must have the same size.
- The naïve version has a tendency to blow up in number of channels.
 - **Why? Max-pooling does not change the number of channels.** When concatenated with other filter responses, number of channels increase with every layer.
 - Solution: Do 1x1 convolution to decrease the number of channels.
 - Also called “bottleneck”.
- In order to decrease the computational complexity of 3x3 and 5x5 pooling, they are also preceded by 1x1 convolution (i.e., the number of channels are reduced).

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as “conv<receptive field size>-<number of channels>”. The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A, A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

ResNet (2015)

- Residual (shortcut) connections

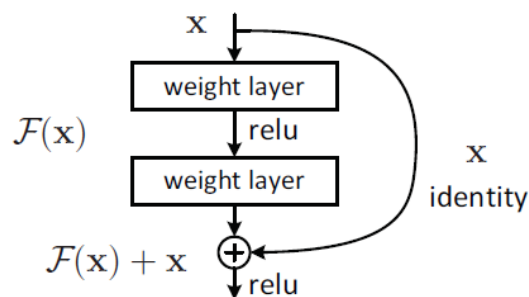


Figure 2. Residual learning: a building block.

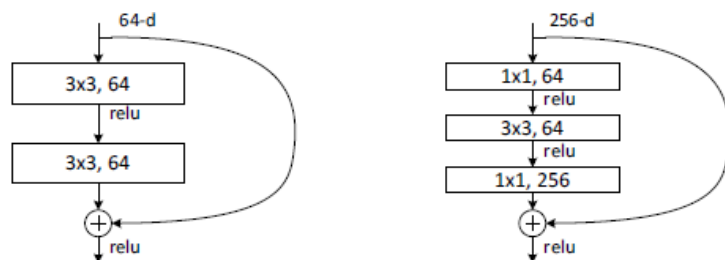
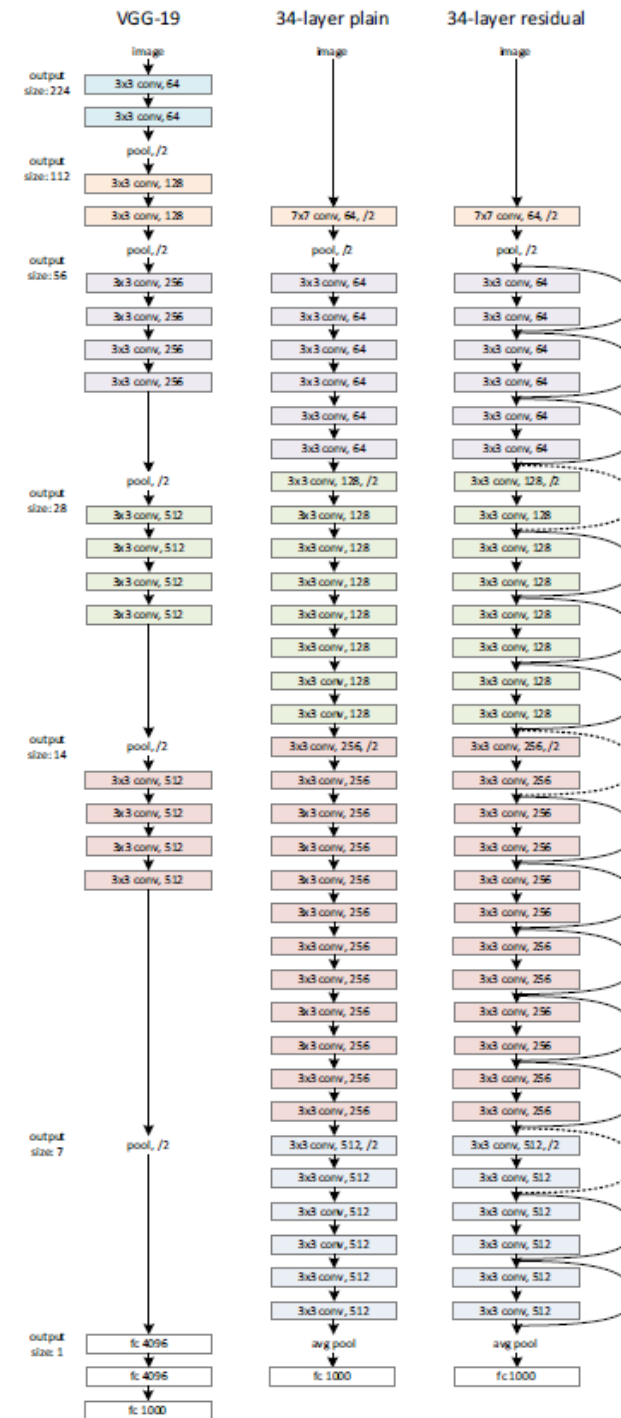


Figure 5. A deeper residual function \mathcal{F} for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.



Effect of residual connections

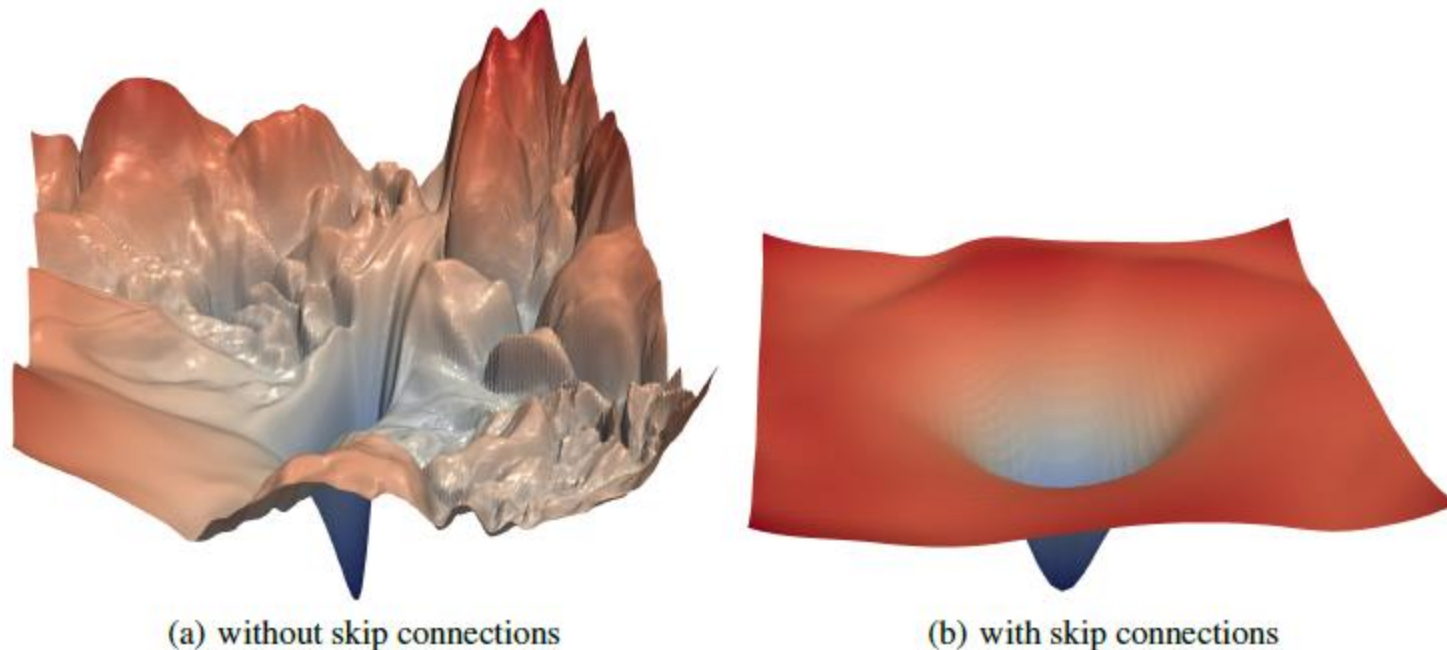


Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The vertical axis is logarithmic to show dynamic range. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

VISUALIZING THE LOSS LANDSCAPE OF NEURAL NETS

2018

Hao Li¹, Zheng Xu¹, Gavin Taylor², Christoph Studer³, Tom Goldstein¹

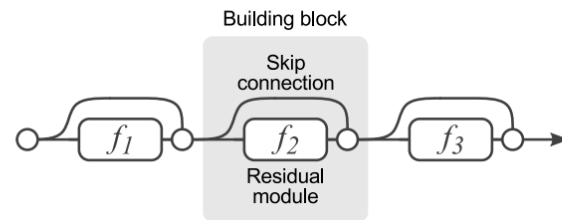
¹University of Maryland, College Park, ²United States Naval Academy, ³Cornell University
{hao.li, zheng.xu, tom.g}@cs.umd.edu, taylor@usna.edu, studer@cornell.edu

ResNet: Ensemble of Shallow Networks

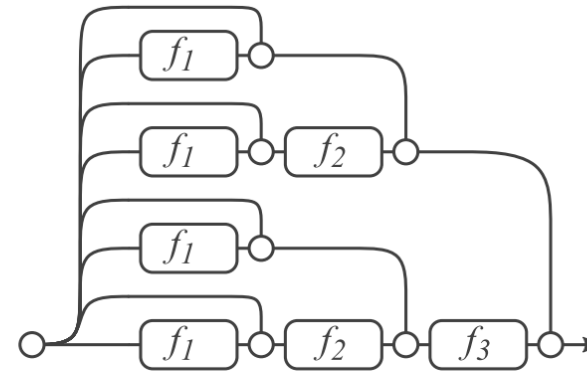
Residual Networks Behave Like Ensembles of Relatively Shallow Networks

Andreas Veit Michael Wilber Serge Belongie
Department of Computer Science & Cornell Tech
Cornell University
{av443, mjlw285, sjb344}@cornell.edu

2016



(a) Conventional 3-block residual network



(b) Unraveled view of (a)

Figure 1: Residual Networks are conventionally shown as (a), which is a natural representation of Equation (1). When we expand this formulation to Equation (6), we obtain an *unraveled view* of a 3-block residual network (b). Circular nodes represent additions. From this view, it is apparent that residual networks have $O(2^n)$ implicit paths connecting input and output and that adding a block doubles the number of paths.

Today

- Convolutional Neural Networks (CNNs)
 - Well-known CNN architectures
- Recurrent Neural Networks (RNNs)

Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10,000-Layer Vanilla Convolutional Neural Networks

Lechao Xiao^{1,2} Yasaman Bahri^{1,2} Jascha Sohl-Dickstein¹ Samuel S. Schoenholz¹ Jeffrey Pennington¹

Abstract

In recent years, state-of-the-art methods in computer vision have utilized increasingly deep convolutional neural network architectures (CNNs), with some of the most successful models employing hundreds or even thousands of layers. A variety of pathologies such as vanishing/exploding gradients make training such deep networks challenging. While residual connections and batch normalization do enable training at these depths, it has remained unclear whether such specialized architecture designs are truly necessary to train deep CNNs. In this work, we demonstrate that it is possible to train vanilla CNNs with ten thousand layers or more simply by using an appropriate initialization scheme. We derive this initialization scheme theoretically by developing a mean field theory for signal propagation and by characterizing the conditions for *dynamical isometry*, the equilibration of singular values of the input-output Jacobian matrix. These conditions require that the convolution operator be an orthogonal transformation in the sense that it is norm-preserving. We present an algorithm for generating such random initial orthogonal convolution kernels and demonstrate empirically that they enable efficient training of extremely deep architectures.

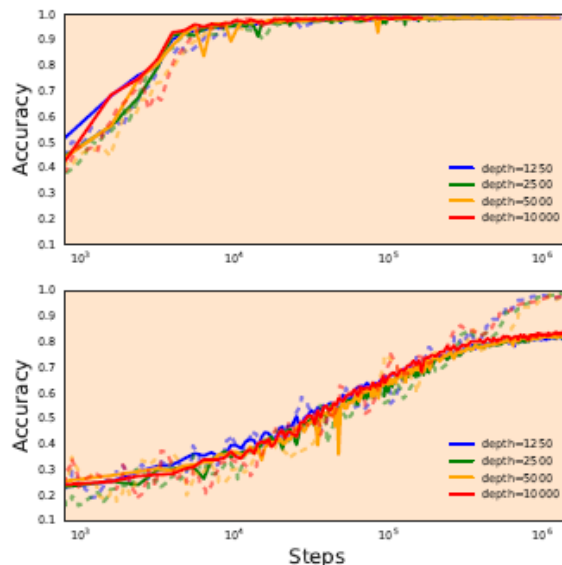


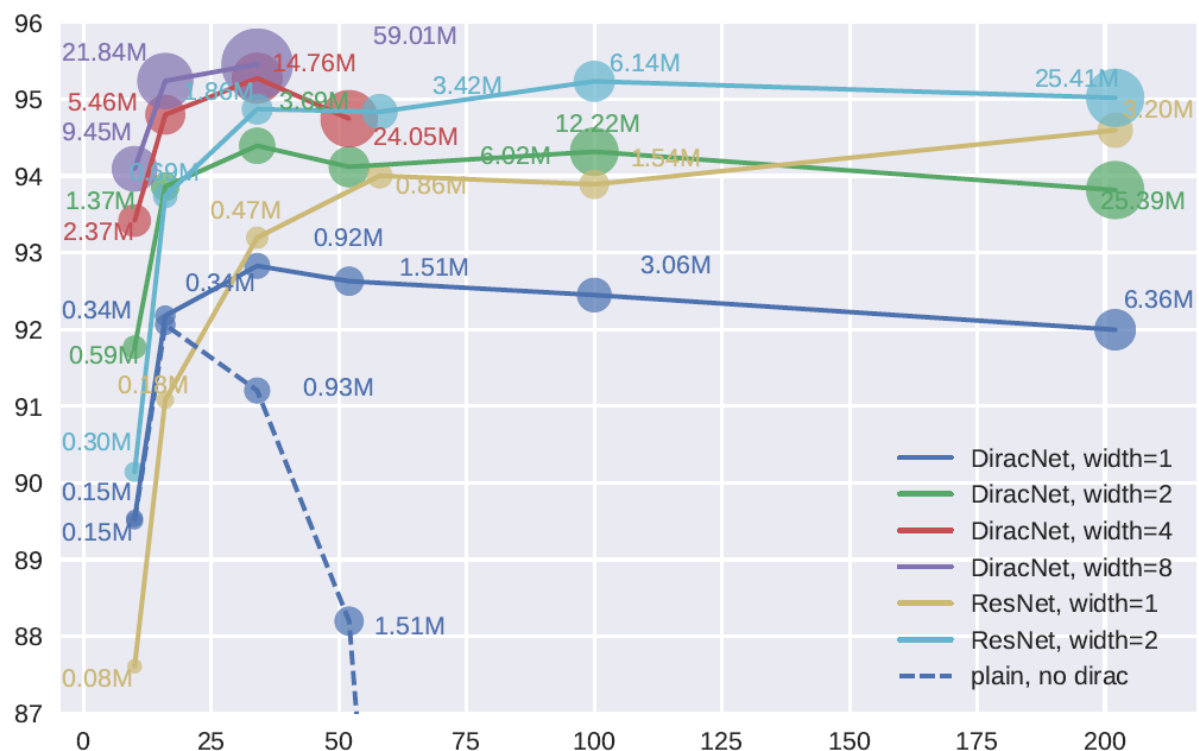
Figure 1. Extremely deep CNNs can be trained without the use of batch normalization or residual connections simply by using a Delta-Orthogonal initialization with critical weight and bias variance and appropriate (in this case, tanh) nonlinearity. Test (solid) and training (dashed) curves on MNIST (top) and CIFAR-10 (bottom) for depths 1,250, 2,500, 5,000, and 10,000.

Kim, 2014), and recently even the board game Go (Silver et al., 2016; 2017).

The performance of deep convolutional networks has improved as these networks have been made ever deeper

"Our initial results suggest that past a certain depth, on the order of tens or hundreds of layers, the test performance for vanilla convolutional architecture saturates. **These observations suggest that architectural features such as residual connections and batch normalization are likely to play an important role in defining a good model class, rather than simply enabling efficient training.**"

DiracNets



SERGEY ZAGORUYKO AND NIKOS KOMODAKIS: DIRACNETS

1

DiracNets: Training Very Deep Neural Networks Without Skip-Connections

Sergey Zagoruyko
sergey.zagoruyko@enpc.fr

Nikos Komodakis
nikos.komodakis@enpc.fr

Université Paris-Est, École des Ponts
ParisTech
Paris, France

Abstract

Deep neural networks with skip-connections, such as ResNet, show excellent performance in various image classification benchmarks. It is though observed that the initial motivation behind them - training deeper networks - does not actually hold true, and the benefits come from increased capacity, rather than from depth. Motivated by this, and inspired from ResNet, we propose a simple Dirac weight parameterization, which allows us to train very deep plain networks without skip-connections, and achieve nearly the same performance. This parameterization has a minor computational cost at training time and no cost at all at inference. We're able to achieve 95.5% accuracy on CIFAR-10 with 34-layer deep plain network, surpassing 1001-layer deep ResNet, and approaching Wide ResNet. Our parameterization also mostly eliminates the need of careful initialization in residual and non-residual networks. The code and models for our experiments are available at <https://github.com/szagoruyko/diracnets>

ResNext

Aggregated Residual Transformations for Deep Neural Networks

Saining Xie¹ Ross Girshick² Piotr Dollár² Zhuowen Tu¹ Kaiming He²
¹UC San Diego ²Facebook AI Research
{s9xie, ztu}@ucsd.edu {rbg, pdollar, kaiminghe}@fb.com **2017**

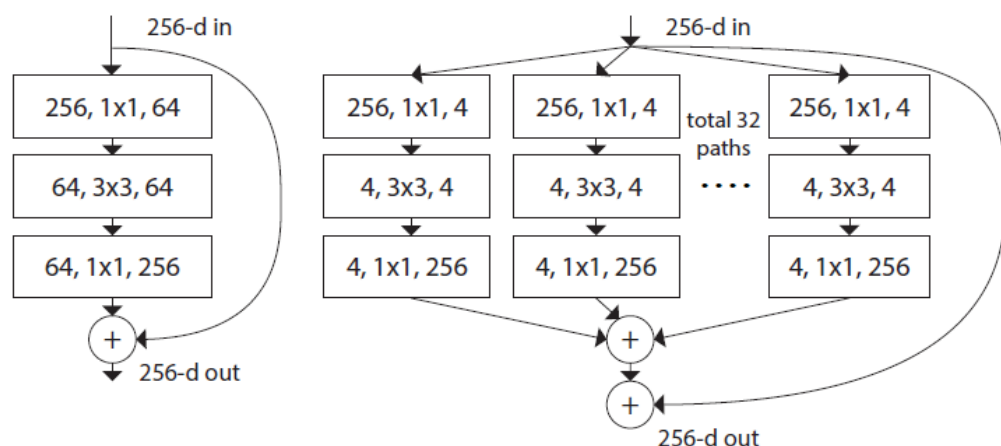


Figure 1. **Left:** A block of ResNet [14]. **Right:** A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

	setting	top-1 err (%)	top-5 err (%)
<i>1 × complexity references:</i>			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
<i>2 × complexity models follow:</i>			
ResNet-200 [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × 100d	21.3	5.7
ResNeXt-101	2 × 64d	20.7	5.5
ResNeXt-101	64 × 4d	20.4	5.3

DenseNet

Densely Connected Convolutional Networks

Gao Huang*
Cornell University
gh349@cornell.edu

Zhuang Liu*
Tsinghua University
liuzhuang13@mails.tsinghua.edu.cn

Laurens van der Maaten
Facebook AI Research
lvdmaaten@fb.com

Kilian Q. Weinberger
Cornell University
kqw4@cornell.edu

2016;2018

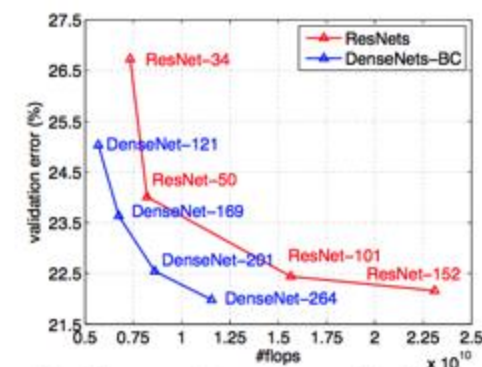
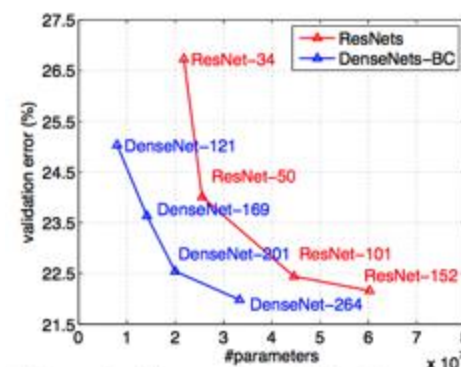
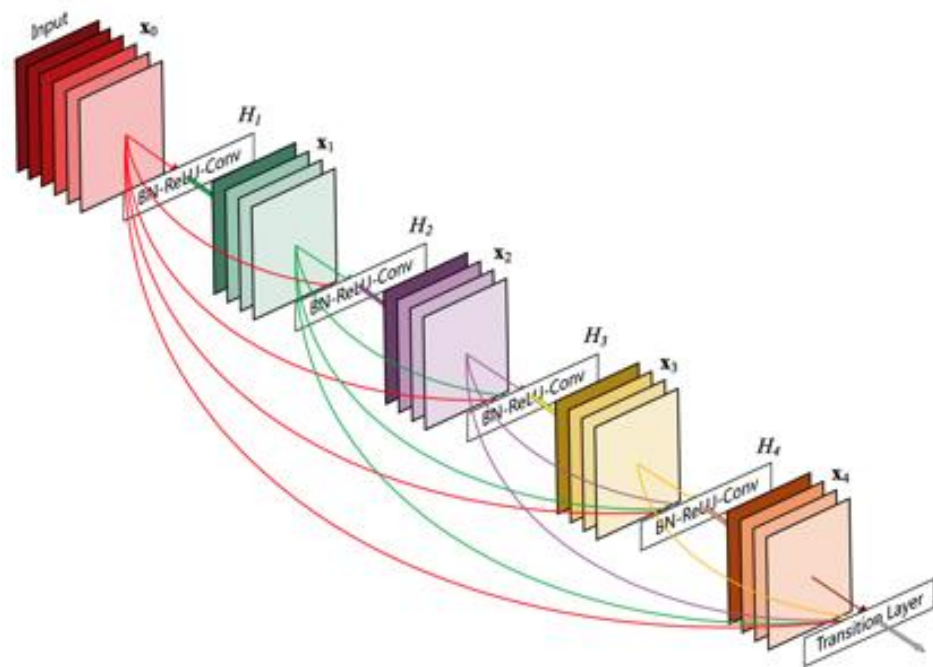


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

Highway networks

- This is a regular MLP with gated units.

Highway Networks

Rupesh Kumar Srivastava
Klaus Greff
Jürgen Schmidhuber

The Swiss AI Lab IDSIA
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
Università della Svizzera italiana (USI)
Scuola universitaria professionale della Svizzera italiana (SUPSI)
Galleria 2, 6928 Manno-Lugano, Switzerland

RUPESH@IDSIA.CH
KLAUS@IDSIA.CH
JUERGEN@IDSIA.CH

Sinan Kalkan

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W}_H). \quad (1)$$

H is usually an affine transform followed by a non-linear activation function, but in general it may take other forms.

For a highway network, we additionally define two non-linear transforms $T(\mathbf{x}, \mathbf{W}_T)$ and $C(\mathbf{x}, \mathbf{W}_C)$ such that

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W}_H) \cdot T(\mathbf{x}, \mathbf{W}_T) + \mathbf{x} \cdot C(\mathbf{x}, \mathbf{W}_C). \quad (2)$$

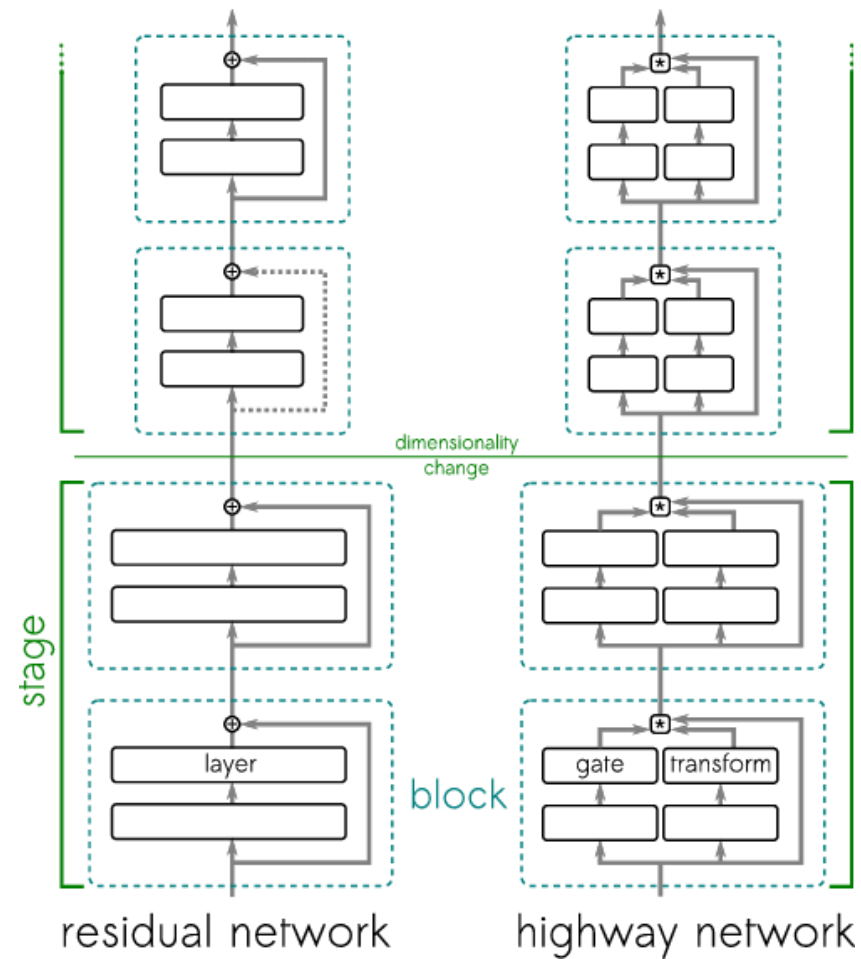
We refer to T as the *transform* gate and C as the *carry* gate, since they express how much of the output is produced by transforming the input and carrying it, respectively. For simplicity, in this paper we set $C = 1 - T$, giving

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W}_H) \cdot T(\mathbf{x}, \mathbf{W}_T) + \mathbf{x} \cdot (1 - T(\mathbf{x}, \mathbf{W}_T)). \quad (3)$$

The dimensionality of $\mathbf{x}, \mathbf{y}, H(\mathbf{x}, \mathbf{W}_H)$ and $T(\mathbf{x}, \mathbf{W}_T)$ must be the same for Equation (3) to be valid. Note that this re-parametrization of the layer transformation is much more flexible than Equation (1). In particular, observe that

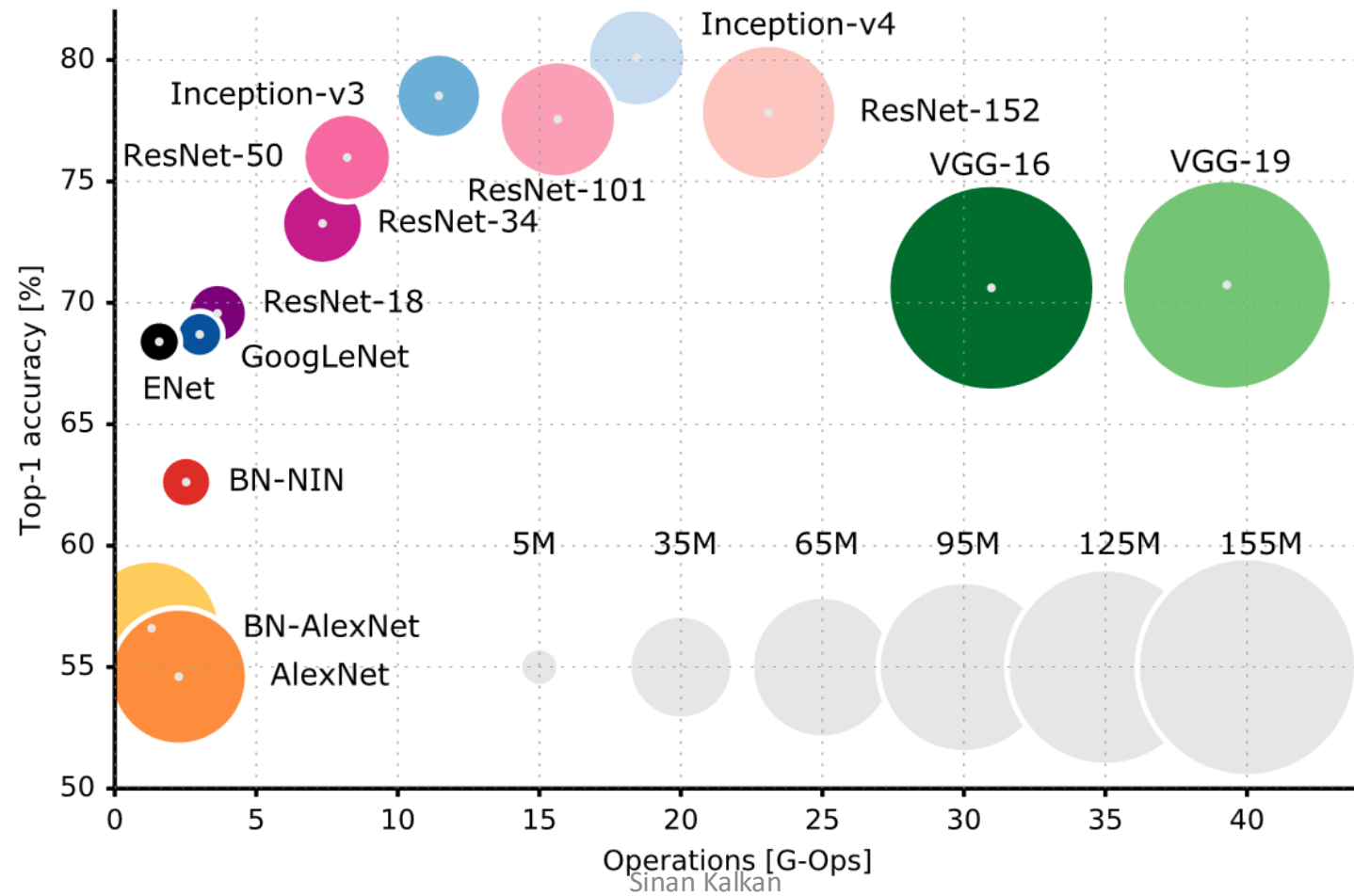
$$\mathbf{y} = \begin{cases} \mathbf{x}, & \text{if } T(\mathbf{x}, \mathbf{W}_T) = 0, \\ H(\mathbf{x}, \mathbf{W}_H), & \text{if } T(\mathbf{x}, \mathbf{W}_T) = 1. \end{cases} \quad (4)$$

Highway Networks



Comparison:

<https://arxiv.org/pdf/1605.07678.pdf>



Going deep may not be the only answer

Shallow Networks for High-Accuracy Road Object-Detection

Khalid Ashraf, Bichen Wu, Forrest N. Iandola, , Matthew W. Moskewicz, Kurt Keutzer
Electrical Engineering and Computer Sciences Department, UC Berkeley

{ashrafkhalid, bichen}@berkeley.edu, {forresti, moskewicz, keutzer}@eecs.berkeley.edu

Abstract

The ability to automatically detect other vehicles on the road is vital to the safety of partially-autonomous and fully-autonomous vehicles. Most of the high-accuracy techniques for this task are based on R-CNN or one of its faster variants. In the research community, much emphasis has been applied to using 3D vision or complex R-CNN variants to achieve higher accuracy. However, are there more straightforward modifications that could deliver higher accuracy? Yes. We show that increasing input image resolution (i.e. upsampling) offers up to 12 percentage-points higher accuracy compared to an off-the-shelf baseline. We also find situations where earlier/shallower layers of CNN provide higher accuracy than later/deeper layers. We further show that shallow models and upsampled images yield competitive accuracy. Our findings contrast with the current trend towards deeper and larger models to achieve high accuracy in domain specific detection tasks.

Recent work

“Towards Principled Design of Deep Convolutional Networks: Introducing SimpNet”

- Major winning Convolutional Neural Networks (CNNs), such as VGGNet, ResNet, DenseNet, \etc, include tens to hundreds of millions of parameters, which impose considerable computation and memory overheads. This limits their practical usage in training and optimizing for real-world applications. On the contrary, light-weight architectures, such as SqueezeNet, are being proposed to address this issue. However, they mainly suffer from low accuracy, as they have compromised between the processing power and efficiency. These inefficiencies mostly stem from following an ad-hoc designing procedure. In this work, we discuss and propose several crucial design principles for an efficient architecture design and elaborate intuitions concerning different aspects of the design procedure. Furthermore, we introduce a new layer called {\it SAF-pooling} to improve the generalization power of the network while keeping it simple by choosing best features. Based on such principles, we propose a simple architecture called {\it SimpNet}. We empirically show that SimpNet provides a good trade-off between the computation/memory efficiency and the accuracy solely based on these primitive but crucial principles. SimpNet outperforms the deeper and more complex architectures such as VGGNet, ResNet, WideResidualNet \etc, on several well-known benchmarks, while having 2 to 25 times fewer number of parameters and operations. We obtain state-of-the-art results (in terms of a balance between the accuracy and the number of involved parameters) on standard datasets, such as CIFAR10, CIFAR100, MNIST and SVHN. The implementations are available at \href{url}{[this https URL](https://arxiv.org/abs/1802.06205)}.

Binary networks

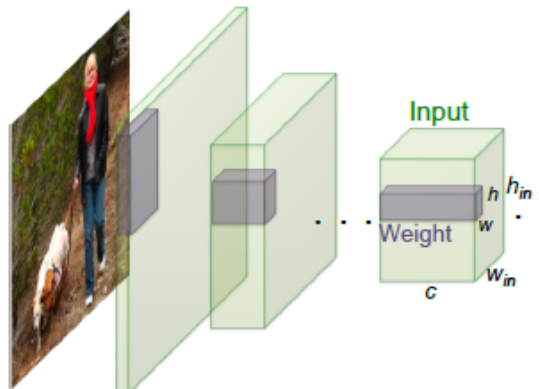
XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks

Mohammad Rastegari[†], Vicente Ordonez[†], Joseph Redmon*, Ali Farhadi^{†*}

Allen Institute for AI[†], University of Washington*
{mohammadr,vicenteor}@allenai.org
{pjreddie,ali}@cs.washington.edu

Abstract. We propose two efficient approximations to standard convolutional neural networks: Binary-Weight-Networks and XNOR-Networks. In Binary-Weight-Networks, the filters are approximated with binary values resulting in $32\times$ memory saving. In XNOR-Networks, both the filters and the input to convolutional layers are binary. XNOR-Networks approximate convolutions using primarily binary operations. This results in $58\times$ faster convolutional operations and $32\times$ memory savings. XNOR-Nets offer the possibility of running state-of-the-art networks on CPUs (rather than GPUs) in real-time. Our binary networks are simple, accurate, efficient, and work on challenging visual tasks. We evaluate our approach on the ImageNet classification task. The classification accuracy with a Binary-Weight-Network version of AlexNet is only 2.9% less than the full-precision AlexNet (in top-1 measure). We compare our method with recent network binarization methods, BinaryConnect and BinaryNets, and outperform these methods by large margins on ImageNet, more than 16% in top-1 accuracy.

Binary networks



	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Time Saving on CPU (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs $\begin{bmatrix} 0.11 & -0.21 & \dots & -0.34 \\ -0.25 & 0.61 & \dots & 0.52 \end{bmatrix}$ Real-Value Weights $\begin{bmatrix} 0.12 & -1.2 & \dots & 0.41 \\ -0.2 & 0.5 & \dots & 0.68 \end{bmatrix}$	$+, -, \times$	1x	1x	%56.7
Binary Weight	Real-Value Inputs $\begin{bmatrix} 0.11 & -0.21 & \dots & -0.34 \\ -0.25 & 0.61 & \dots & 0.52 \end{bmatrix}$ Binary Weights $\begin{bmatrix} 1 & -1 & \dots & 1 \\ -1 & 1 & \dots & 1 \end{bmatrix}$	$+, -$	$\sim 32x$	$\sim 2x$	%53.8
BinaryWeight Binary Input (XNOR-Net)	Binary Inputs $\begin{bmatrix} 1 & -1 & \dots & -1 \\ -1 & 1 & \dots & 1 \end{bmatrix}$ Binary Weights $\begin{bmatrix} 1 & -1 & \dots & 1 \\ -1 & 1 & \dots & 1 \end{bmatrix}$	XNOR, bitcount	$\sim 32x$	$\sim 58x$	%44.2

Fig. 1: We propose two efficient variations of convolutional neural networks. **Binary-Weight-Networks**, when the weight filters contains binary values. **XNOR-Networks**, when both weigh and input have binary values. These networks are very efficient in terms of memory and computation, while being very accurate in natural image classification. This offers the possibility of using accurate vision techniques in portable devices with limited resources.

Exploring Randomly Wired Neural Networks for Image Recognition

Saining Xie Alexander Kirillov Ross Girshick Kaiming He

Facebook AI Research (FAIR)

Abstract

Neural networks for image recognition have evolved through extensive manual design from simple chain-like models to structures with multiple wiring paths. The success of ResNets [11] and DenseNets [16] is due in large part to their innovative wiring plans. Now, neural architecture search (NAS) studies are exploring the joint optimization of wiring and operation types, however, the space of possible wirings is constrained and still driven by manual design despite being searched. In this paper, we explore a more diverse set of connectivity patterns through the lens of randomly wired neural networks. To do this, we first define the concept of a stochastic network generator that encapsulates the entire network generation process. Encapsulation provides a unified view of NAS and randomly wired networks. Then, we use three classical random graph models to generate randomly wired graphs for networks. The results are surprising: several variants of these random generators yield network instances that have competitive accuracy on the ImageNet benchmark. These results suggest that new efforts focusing on designing better network generators may lead to new breakthroughs by exploring less constrained search spaces with more room for novel design.

1 Introduction

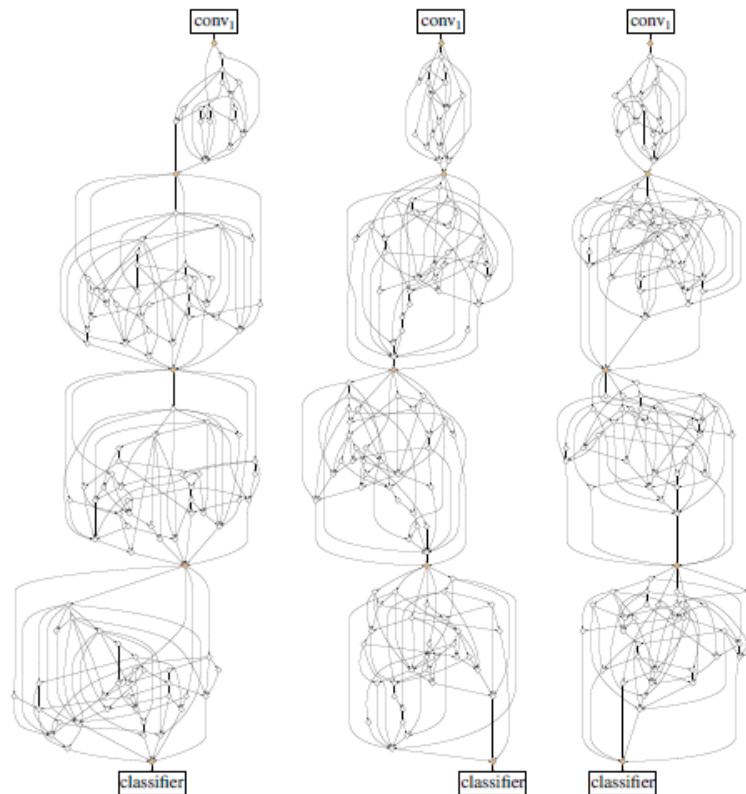


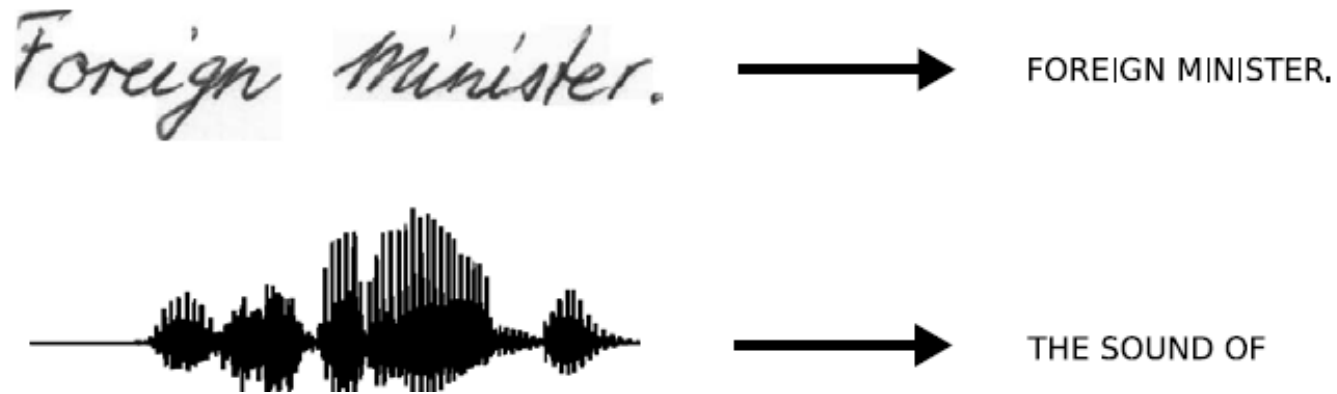
Figure 1. Randomly wired neural networks generated by the classical Watts-Strogatz (WS) [50] model: these three instances of random networks achieve (left-to-right) 79.1%, 79.1%, 79.0% classification accuracy on ImageNet under a similar computational budget to ResNet-50, which has 77.1% accuracy.

CNNs: Summary & future directions

- Less parameters
- Allows going deeper
- High flexibility
 - In operations
 - In organization of layers
 - In the overall architecture etc.
- Future directions:
 - Understanding them better
 - Making them deeper, faster and more efficient
 - Compressing a big network into a smaller & cheaper one.
 - ...

Sequence Labeling/Modeling: Motivation

Why do we need them?



Different types of sequence learning / recognition problems

- Sequence Classification
 - A sequence to a label
 - E.g., recognizing a single spoken word
 - Length of the sequence is fixed
 - Why RNNs then? Because sequential modeling provides robustness against translations and distortions.
- Segment Classification
 - Segments in a sequence correspond to labels
- Temporal Classification
 - General case: sequence (input) to sequence (label) modeling.
 - May not have clue about where input or label starts.

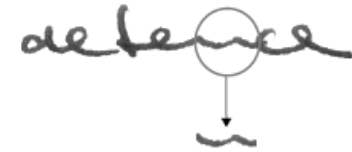
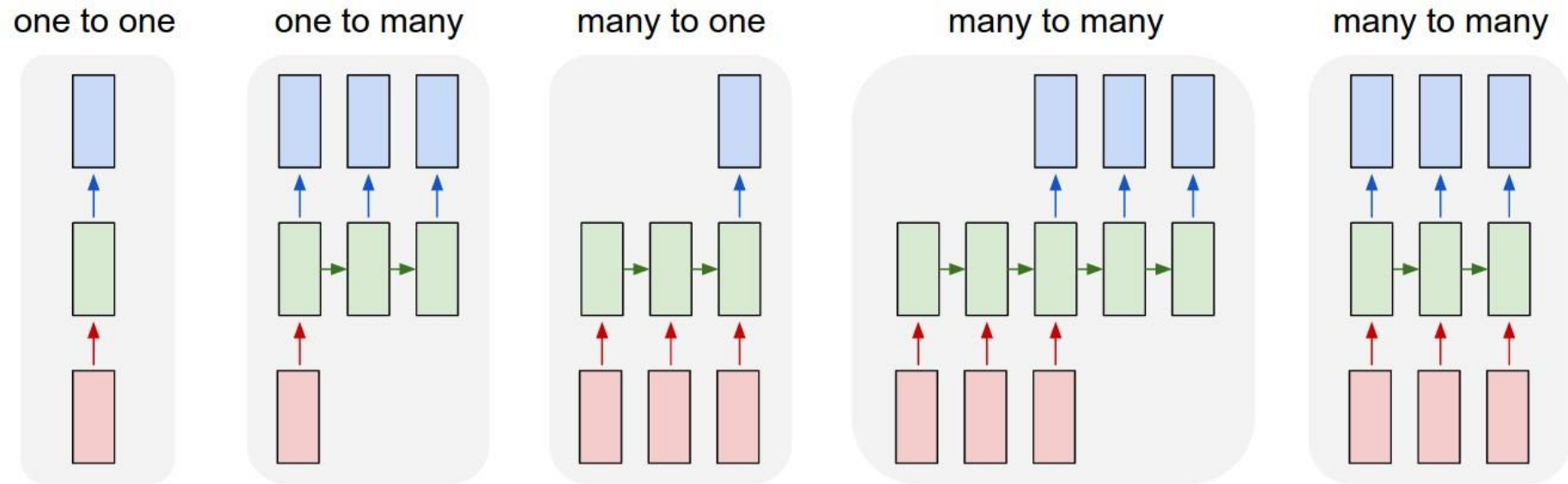


Fig. 2.3 Importance of context in segment classification. The word 'defence' is clearly legible. However the letter 'n' in isolation is ambiguous.

A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks", 2012.

Different types of sequence learning / recognition problems



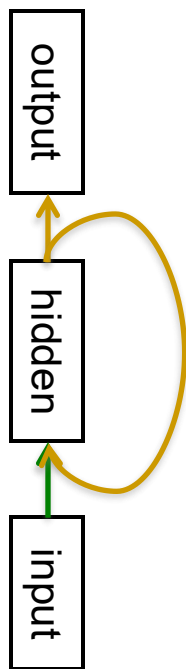
<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

Recurrent Neural Networks

Recurrent Neural Networks (RNNs)



Feed-forward
networks

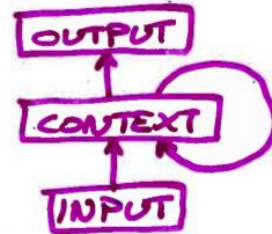


Recurrent
networks

- RNNs are very powerful because:
 - Distributed hidden state that allows them to store a lot of information about the past efficiently.
 - Non-linear dynamics that allows them to update their hidden state in complicated ways.
- With enough neurons and time, RNNs can compute anything that can be computed by your computer.
- More formally, **RNNs are Turing complete.**

Some examples

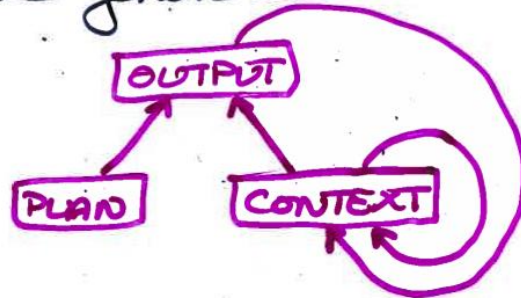
- Temporal pattern recognition



$INPUT_1, INPUT_2, INPUT_3, \dots, INPUT_t$
 $\Rightarrow OUTPUT_t$

e.g., speech recognition
e.g., event recognition
e.g., natural language understanding

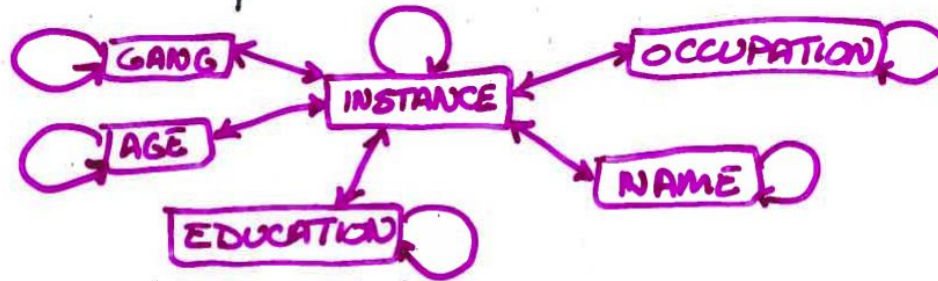
- Sequence generation



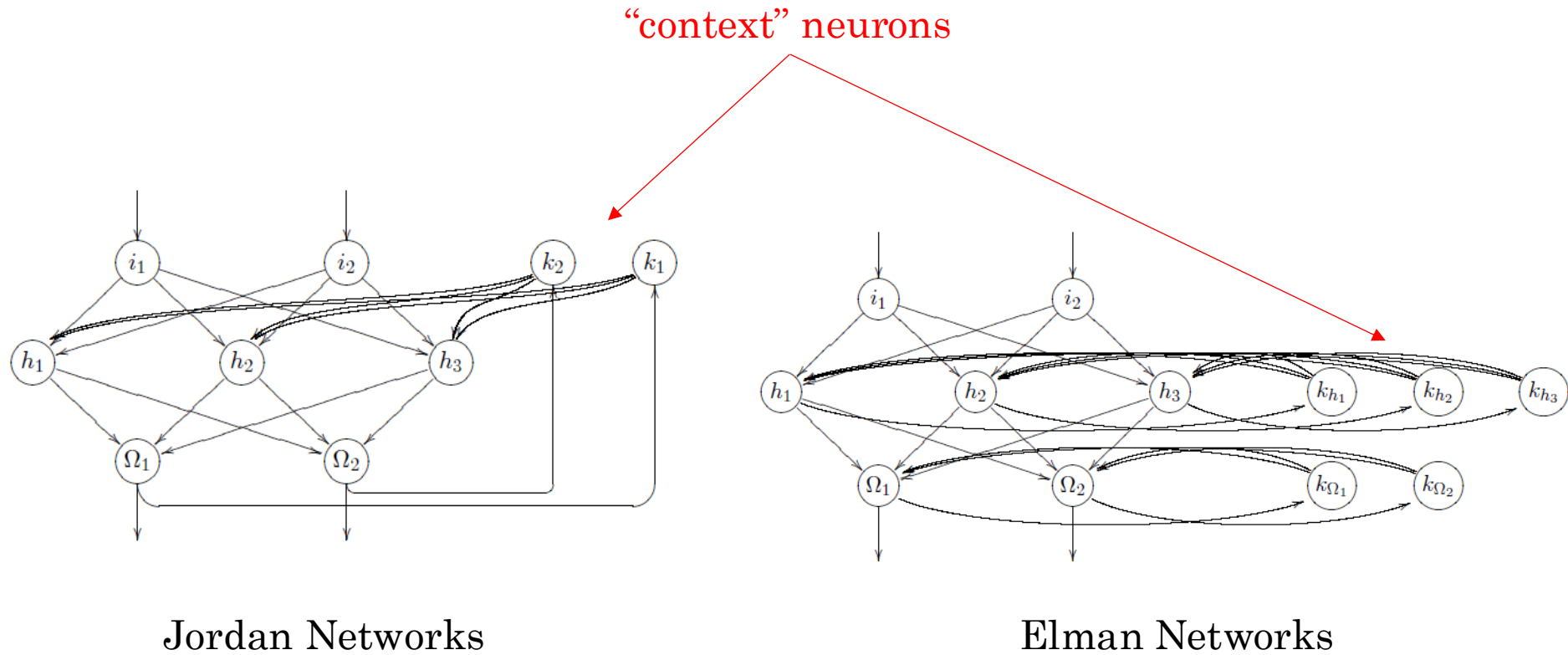
$PLAN \Rightarrow OUTPUT_1, OUTPUT_2,$
 $OUTPUT_3, \dots, OUTPUT_t$

e.g., speech production
e.g., motor control
e.g., planning and acting

- Pattern completion / constraint satisfaction



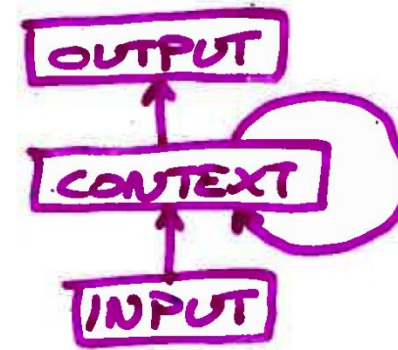
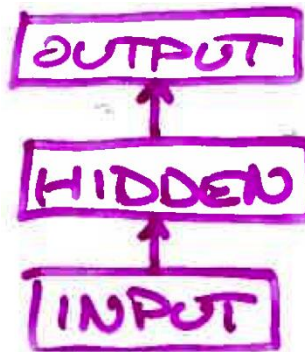
Some examples



Figs: David Kriesel

Challenge

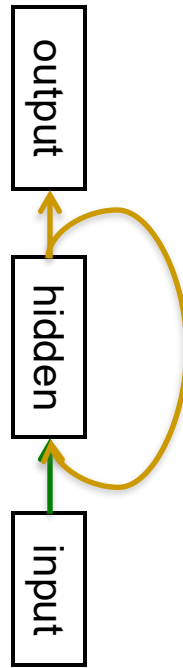
- Back propagation is designed for feedforward nets
- What would it mean to back propagate through a recurrent network?
 - error signal would have to travel back in time



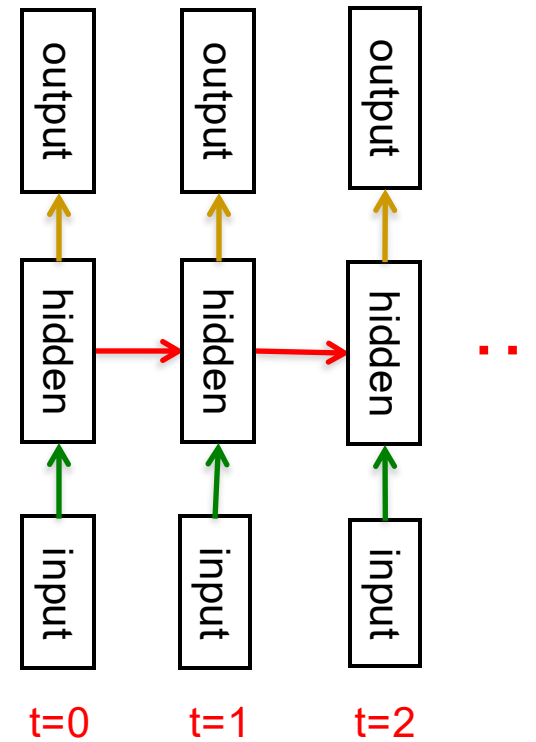
Unfolding



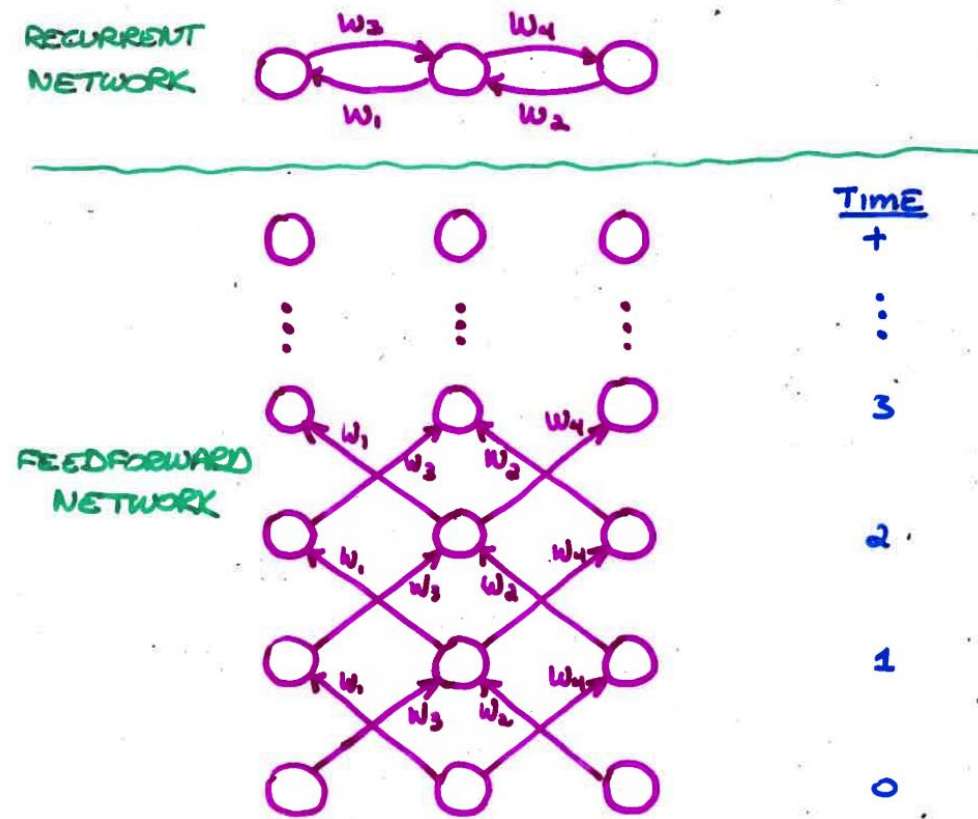
Feed-forward
networks



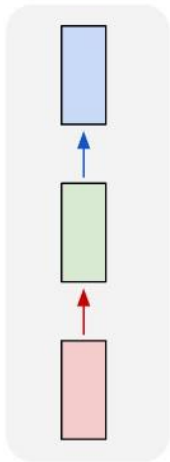
Recurrent
networks



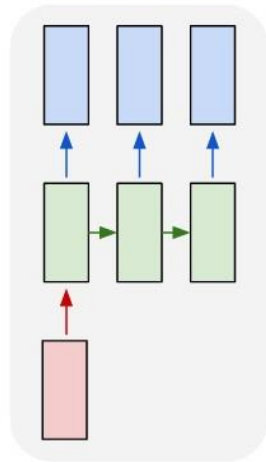
Unfolding (another example)



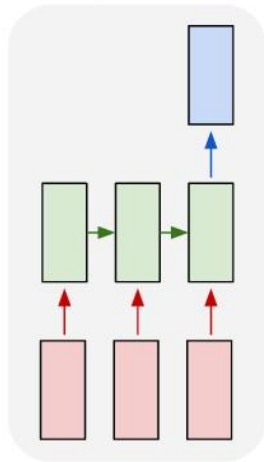
one to one



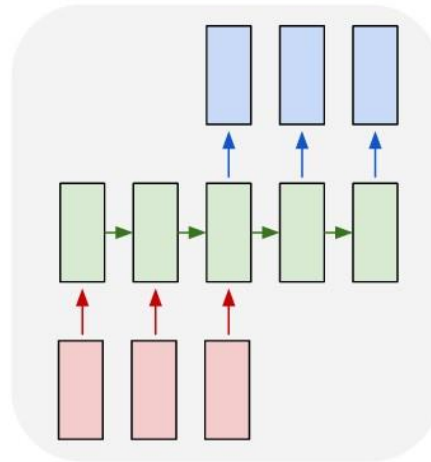
one to many



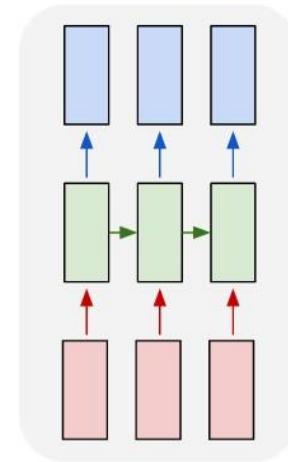
many to one



many to many

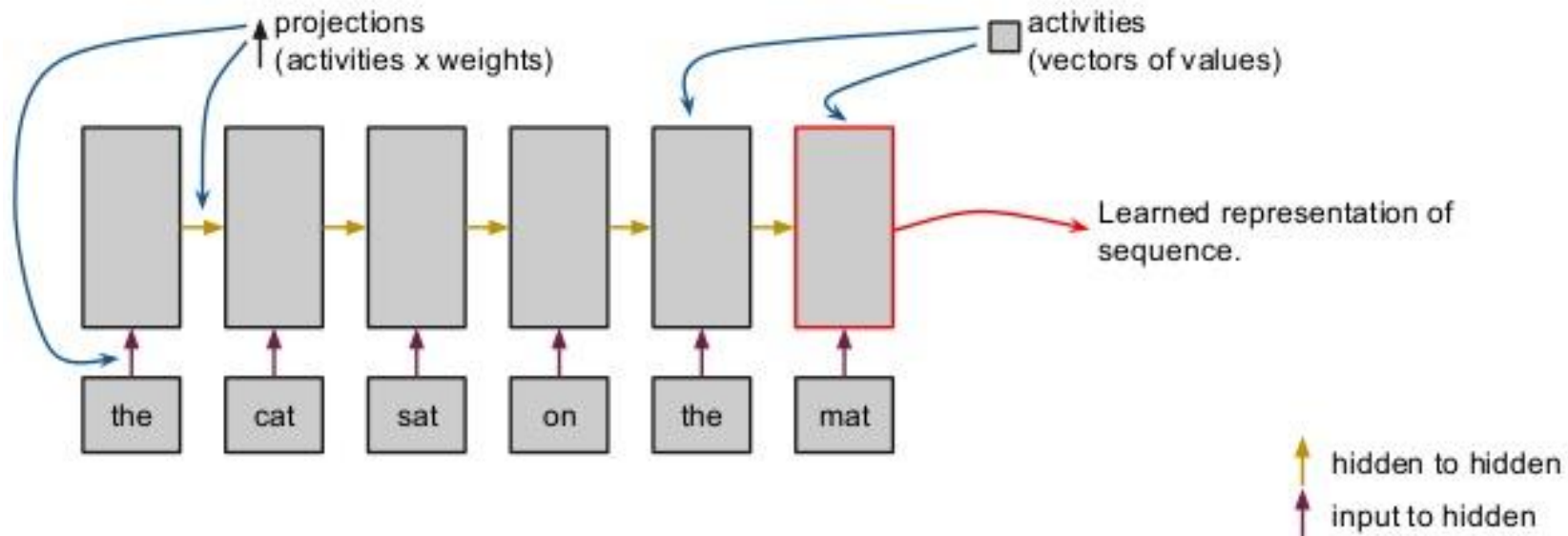


many to many

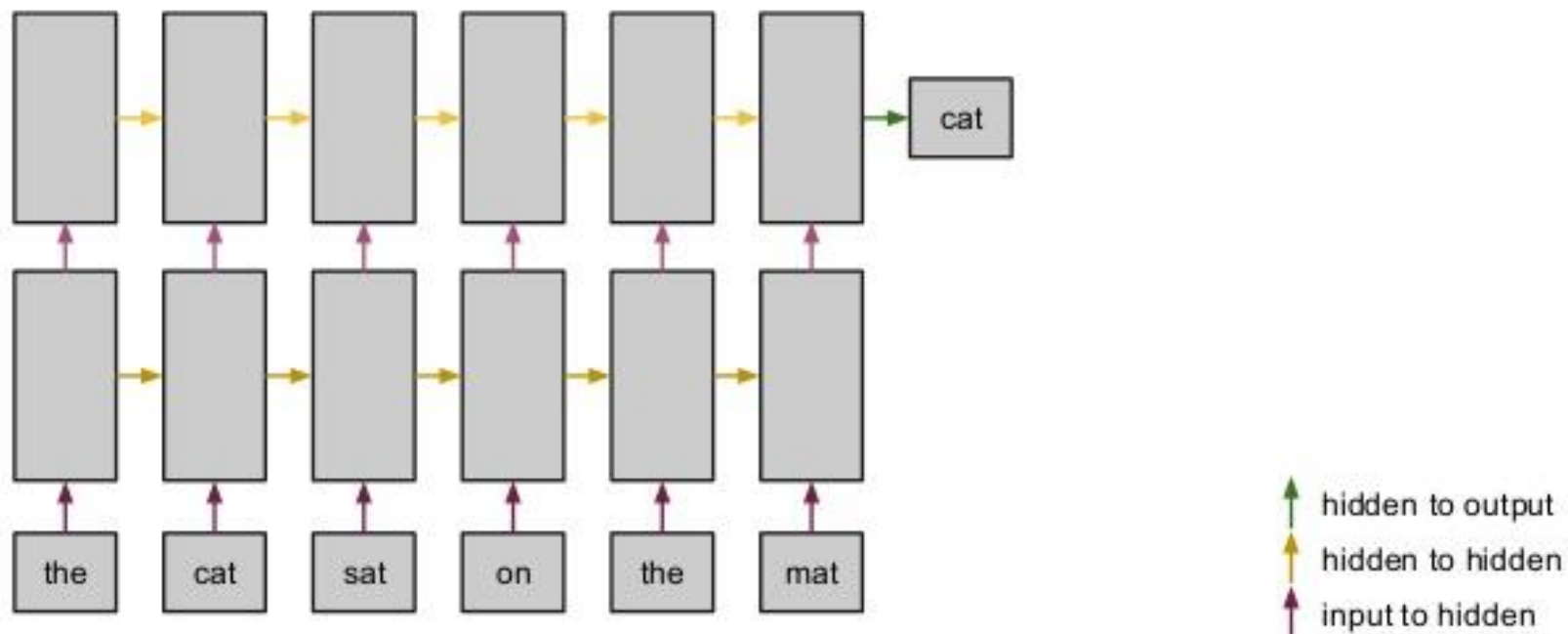


<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

How an RNN works



You can stack them too



Unfolding implications

- Entails duplication of weights => weight sharing
- Sharing weights means their gradients will be accumulated over time and reflected on the weights
- Unfolded network has the **same dynamics of the RNN** for a **fixed number of time steps!**