Lesson 7

PRACTICAL DEEP LEARNING FOR CODERS 2022

WHAT'S INSIDE A NEURAL NET?

Entity Embeddings of Categorical Variables

Cheng Guo[*] and Felix Berkhahn[†]
Neokami Inc.
(Dated: April 25, 2016)

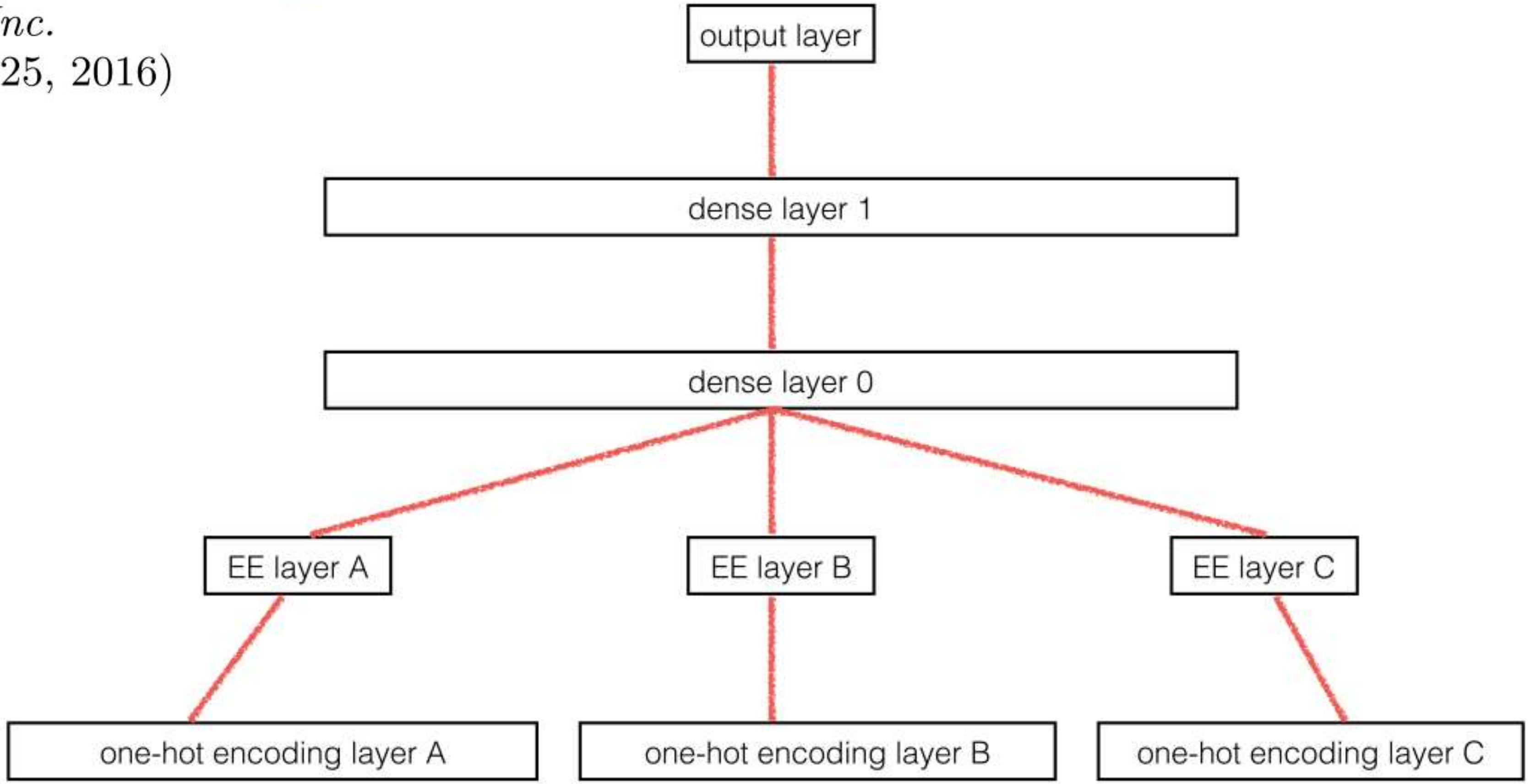
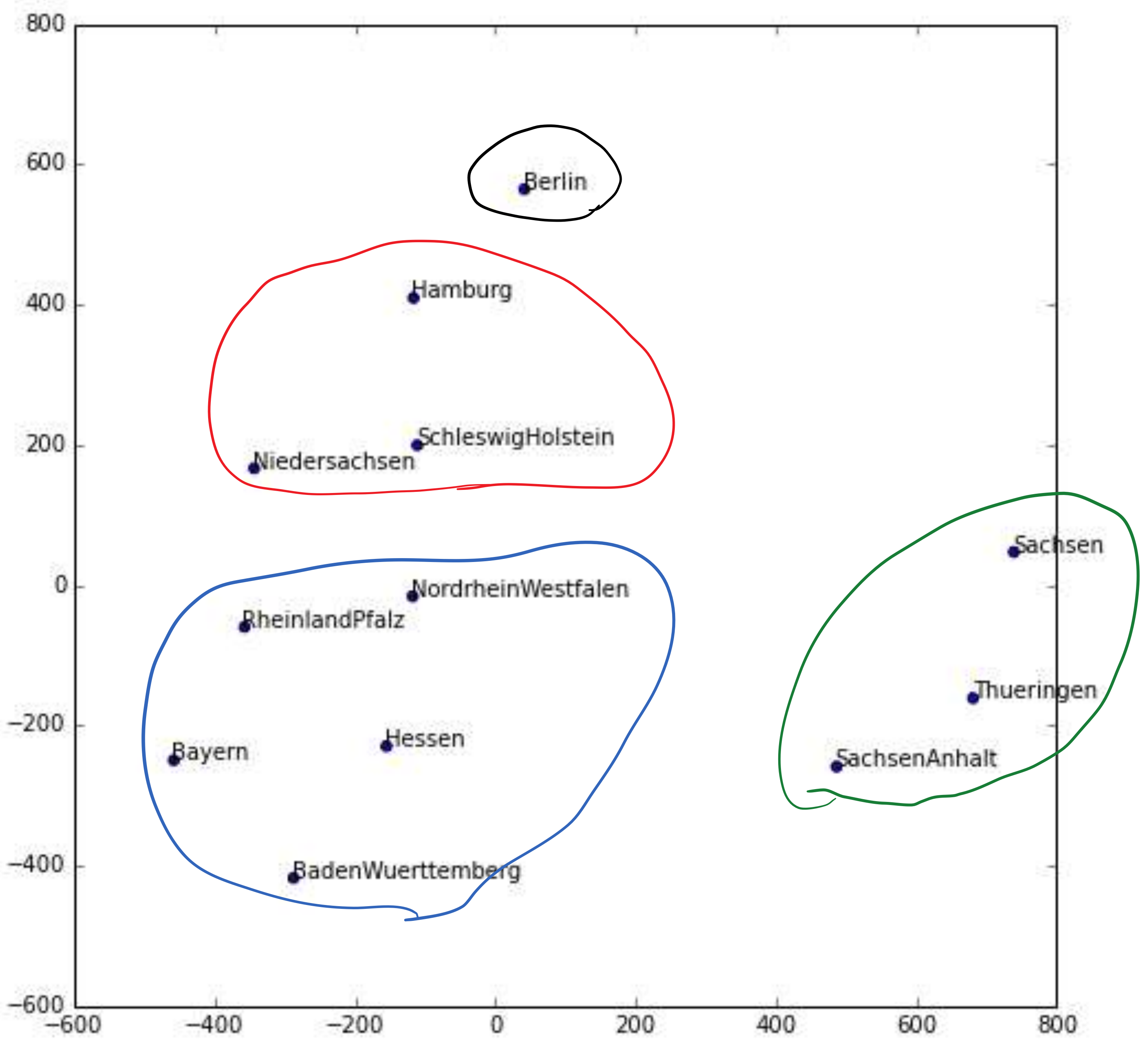
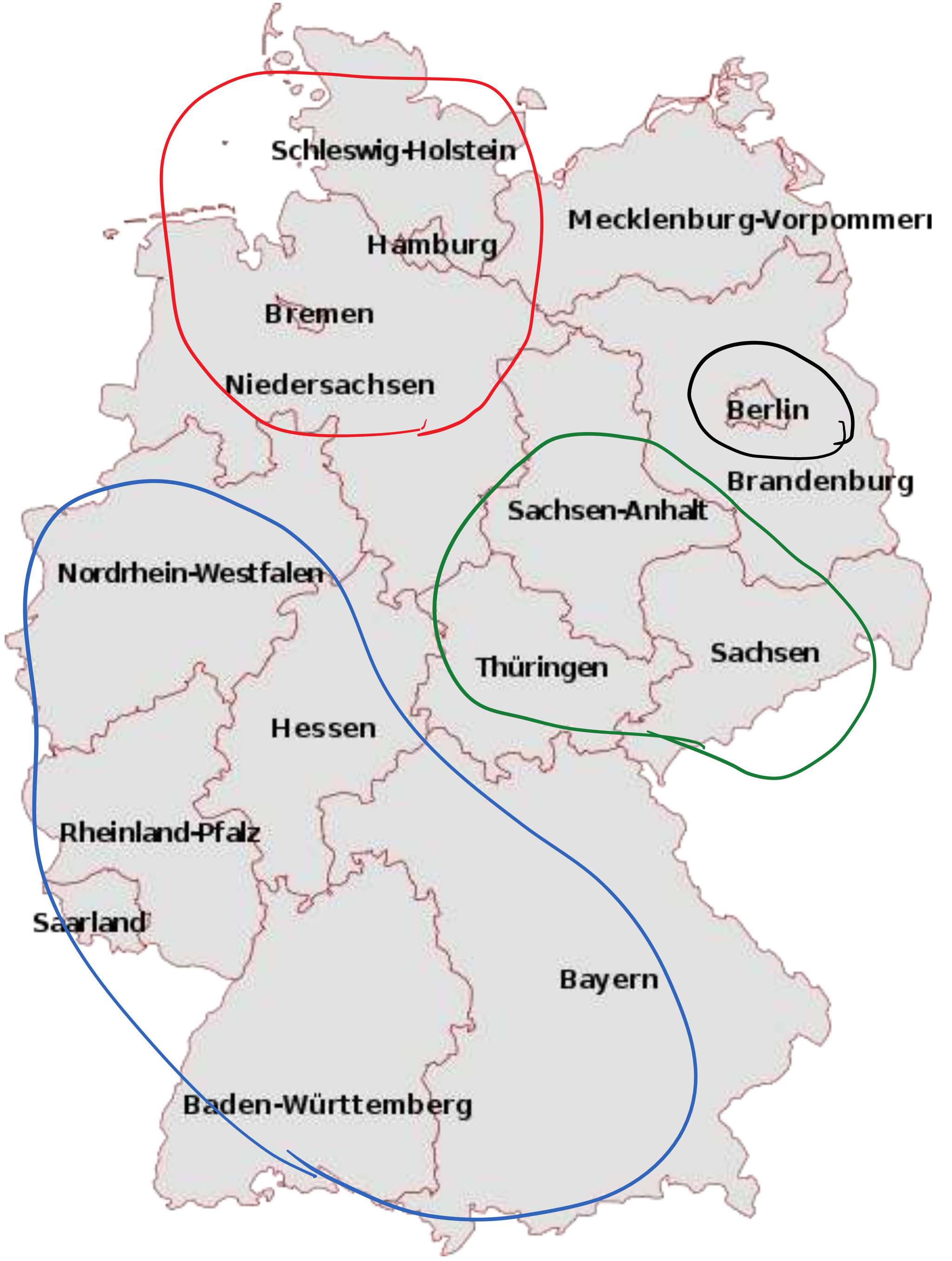
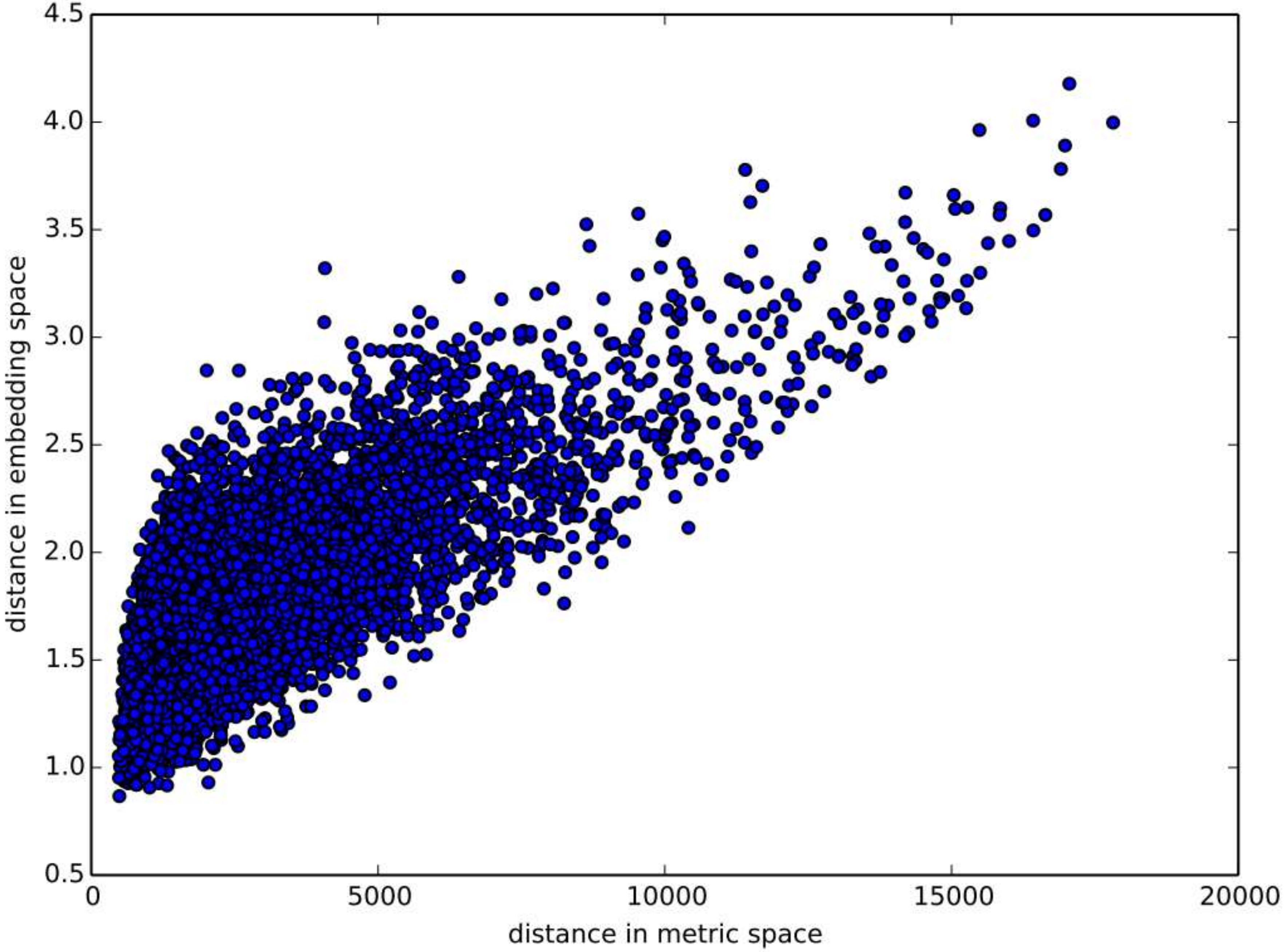


FIG. 1. Illustration that entity embedding layers are equivalent to extra layers on top of each one-hot encoded input.

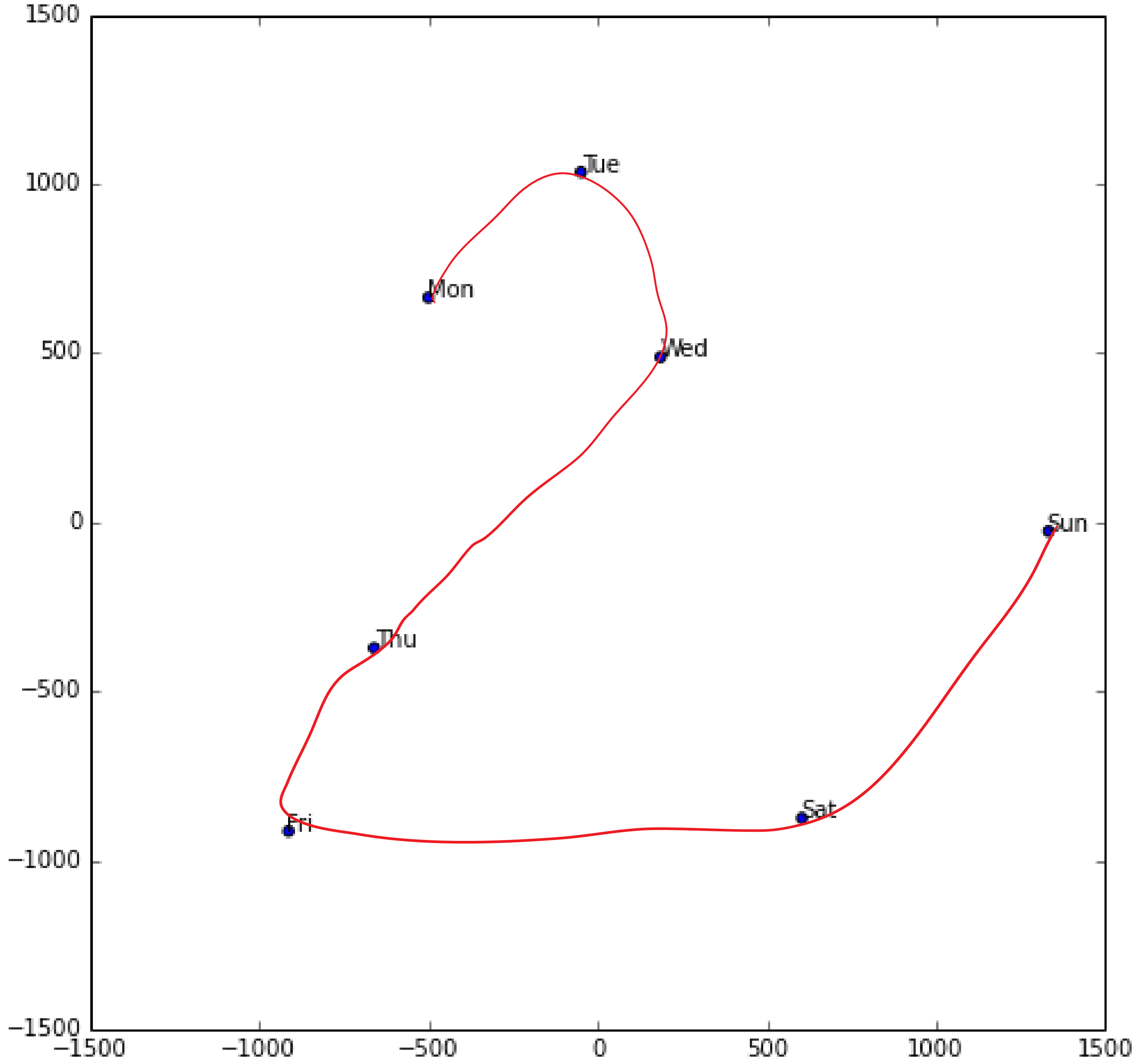
method	MAPE	MAPE (with EE)
KNN	0.290	0.116
random forest	0.158	0.108
gradient boosted trees	0.152	0.115
neural network	0.101	0.093


800
600
400
200
0
−200
−400
−600
−600
−400
−200
0
200
400
600
800
Berlin
Hamburg
SchleswigHolstein
Niedersachsen
Sachsen
NordrheinWestfalen
RheinlandPfalz
Thueringen
Bayern
Hessen
SachsenAnhalt
BadenWuerttemberg

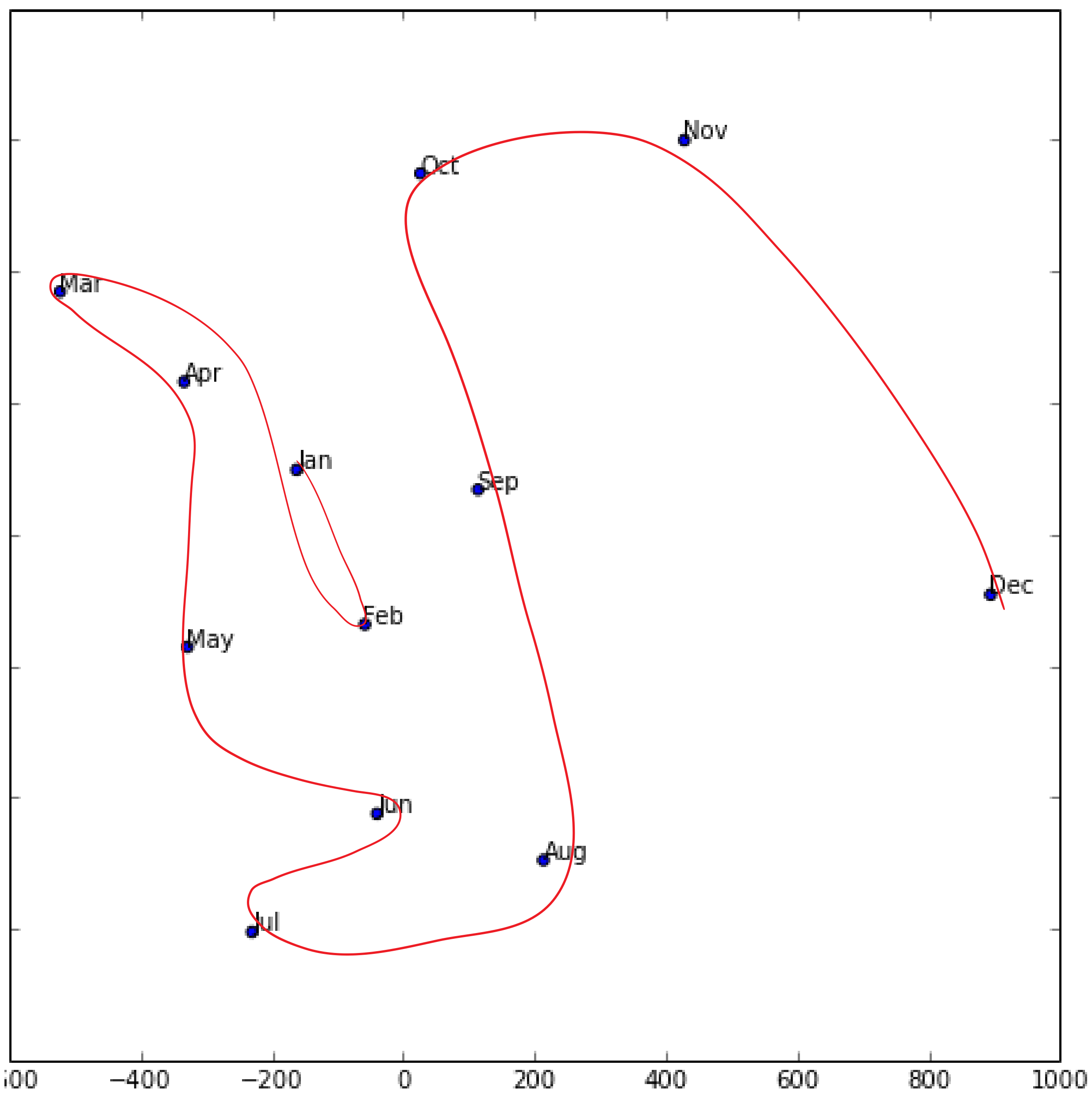

Schleswig-Holstein
Hamburg
Mecklenburg-Vorpommern
Bremen
Niedersachsen
Berlin
Brandenburg
Sachsen-Anhalt
Nordrhein-Westfalen
Thüringen
Sachsen
Hessen
Rheinland-Pfalz
Saarland
Bayern
Baden-Württemberg



4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
distance in embedding space
0
5000
10000
15000
20000
distance in metric space



1500
1000
500
0
−500
−1000
−1500
−1500
−1000
−500
0
500
1000
1500
Tue
Mon
Wed
Sun
Thu
Fri
Sat



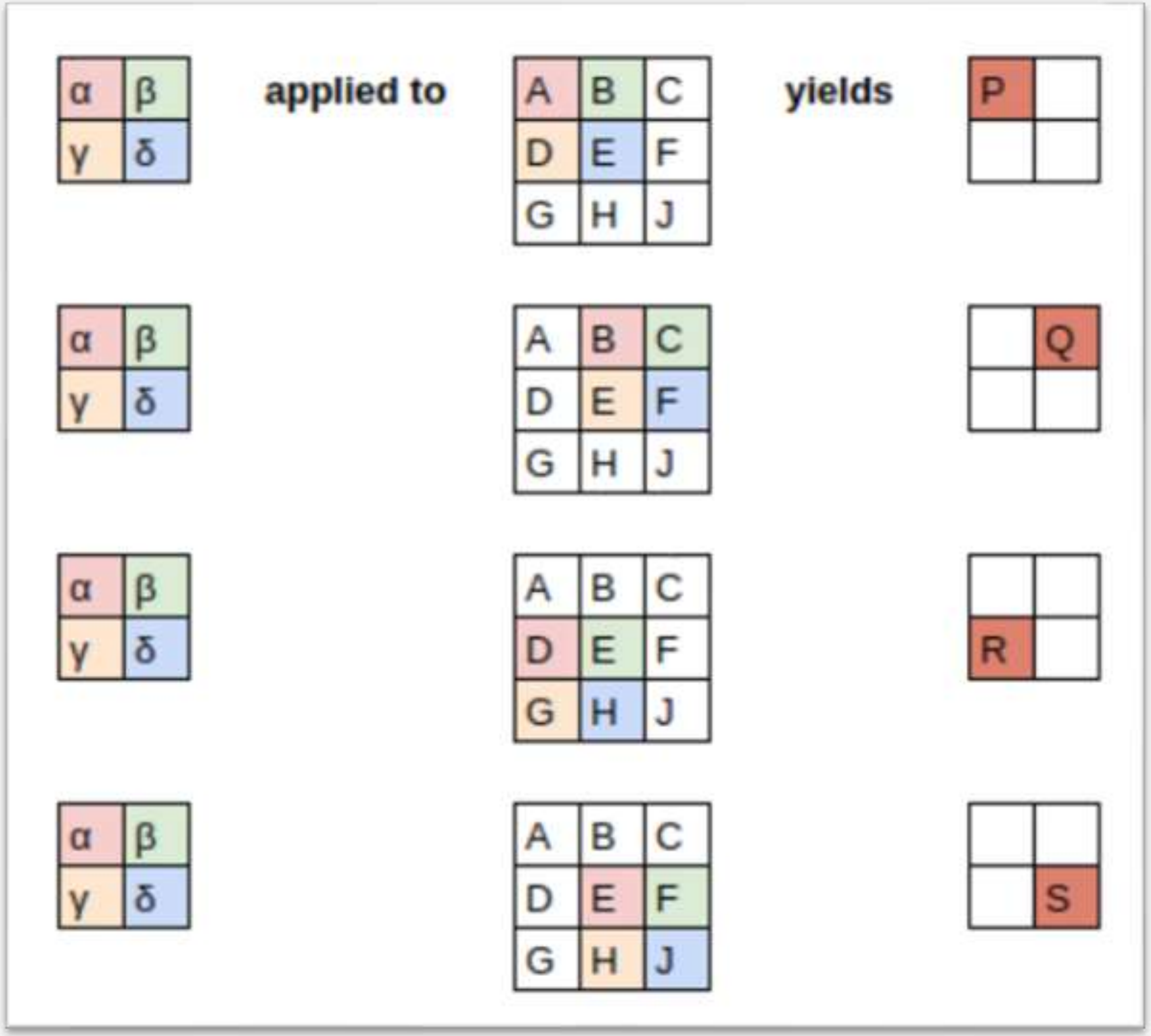
−400
−200
0
200
400
600
800
1000
Nov
Oct
Mar
Apr
Jan
Sep
Dec
Feb
May
Jun
Aug
Jul

Matthew Kleinsmith Follow
Studying deep learning
Feb 10, 2017 · 4 min read

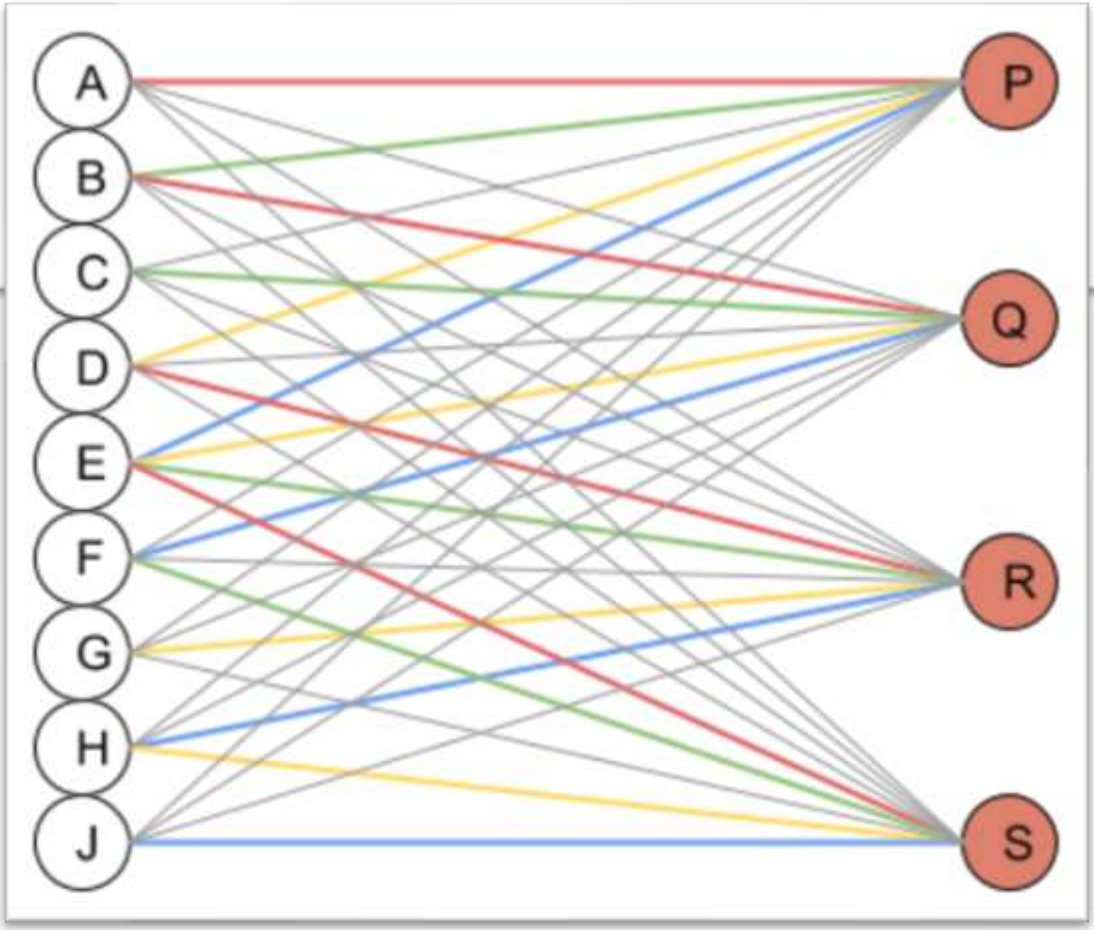
CNNs from different viewpoints

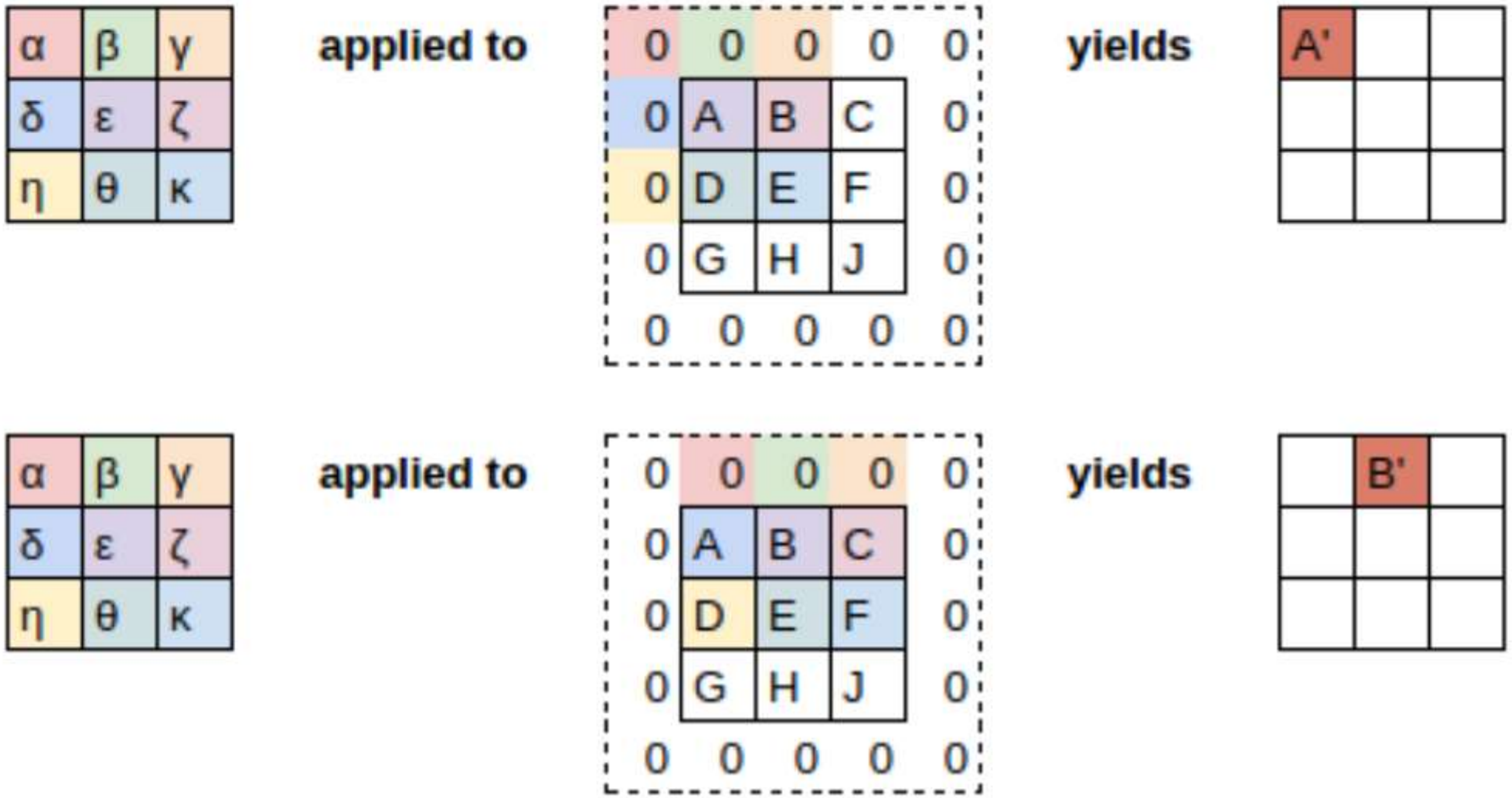
Prerequisite: Basic neural networks

$$\alpha A+\beta B+\gamma D+\delta E+b = P$$
$$\alpha B+\beta C+\gamma E+\delta F+b = Q$$
$$\alpha D+\beta E+\gamma G+\delta H+b = R$$
$$\alpha E+\beta F+\gamma H+\delta J+b = S$$



$$\begin{array}{c}\begin{bmatrix} \alpha & \beta & 0 & \gamma & \delta & 0 & 0 & 0 & 0 \\ 0 & \alpha & \beta & 0 & \gamma & \delta & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha & \beta & 0 & \gamma & \delta & 0 \\ 0 & 0 & 0 & 0 & \alpha & \beta & 0 & \gamma & \delta \end{bmatrix} \\ \begin{matrix} A & B & C & D & E & F & G & H & J \end{matrix}\end{array} * \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ J \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \\ b \end{bmatrix} = \begin{bmatrix} \alpha A+\beta B+0C+\gamma D+\delta E+0F+0G+0H+0J+b \\ 0A+\alpha B+\beta C+0D+\gamma E+\delta F+0G+0H+0J+b \\ 0A+0B+0C+\alpha D+\beta E+0F+\gamma G+\delta H+0J+b \\ 0A+0B+0C+0D+\alpha E+\beta F+0G+\gamma H+\delta J+b \end{bmatrix} = \begin{bmatrix} \alpha A+\beta B+\gamma D+\delta E+b \\ \alpha B+\beta C+\gamma E+\delta F+b \\ \alpha D+\beta E+\gamma G+\delta H+b \\ \alpha E+\beta F+\gamma H+\delta J+b \end{bmatrix} = \begin{bmatrix} P \\ Q \\ R \\ S \end{bmatrix}$$




α β γ
δ ε ζ
η θ κ
applied to
0 0 0 0 0
0 A B C 0
0 D E F 0
0 G H J 0
0 0 0 0 0
yields
A'
α β γ
δ ε ζ
η θ κ
applied to
0 0 0 0 0
0 A B C 0
0 D E F 0
0 G H J 0
0 0 0 0 0
yields
B'

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	
SoftPlus		$f(x) = \log_e(1 + e^x)$	

SAGAR SHARMA Follow

I am interested in Programming (Python, C++), Arduino, Machine learning :) I'm the editor of Arduino Community on Medium. I also like to write stuff.

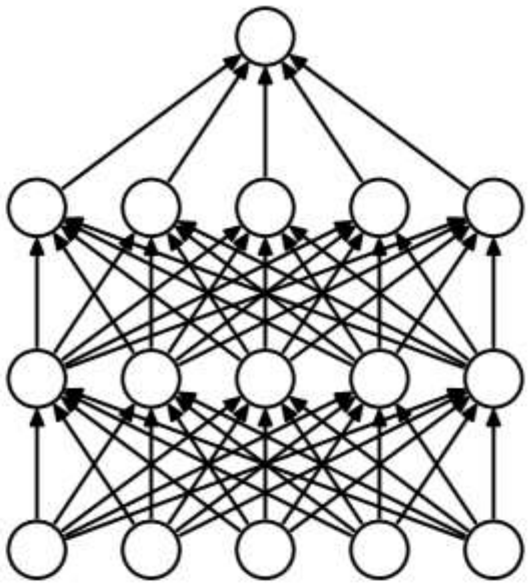
Sep 6, 2017 · 5 min read

Activation Functions: Neural Networks

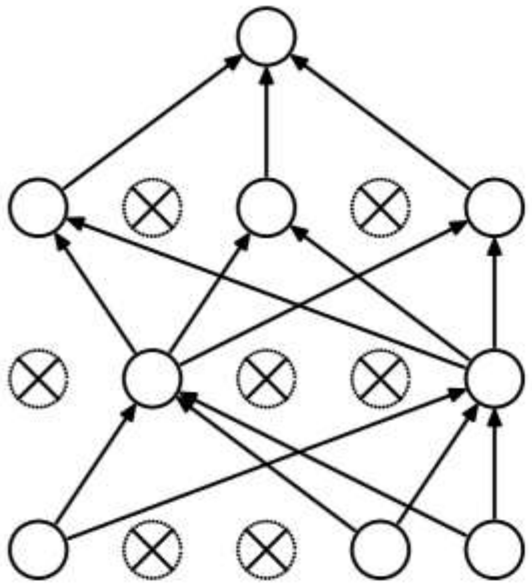
Sigmoid, tanh, Softmax, ReLU, Leaky ReLU EXPLAINED !!!

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava NITISH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
Alex Krizhevsky KRIZ@CS.TORONTO.EDU
Ilya Sutskever ILYA@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU



(a) Standard Neural Net



(b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

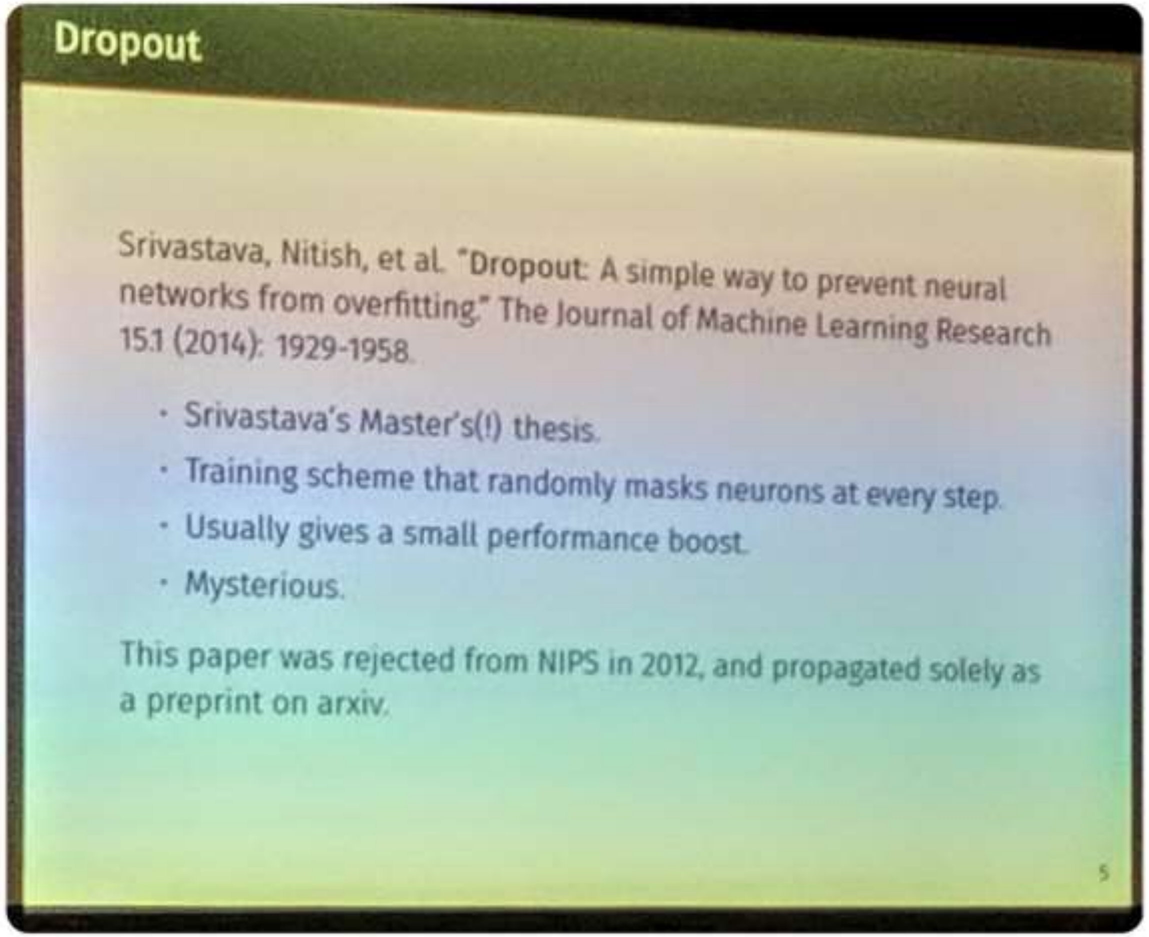
"I went to my bank. The tellers kept changing and I asked one of them why. He said he didn't know but they got moved around a lot. I figured it must be because it would require cooperation between employees to successfully defraud the bank. This made me realize that randomly removing a different subset of neurons on each example would prevent conspiracies and thus reduce overfitting"

Hinton: Reddit AMA

Chris Gorgolewski
@ChrisFiloG

Follow

Did you know that Dropout was originally introduced in a Master's thesis and was rejected from NIPS? Was disseminated via #arxiv! #OHBM2018



5:30 PM - 20 Jun 2018

So what now? Watch the videos again, and...

- **Write**
  - Code
  - Papers
  - Blog Posts
- **Help**
  - Forum
  - Success stories
- **Gather**
  - Book club
  - Meetups
  - Study groups
- **Build**
  - Apps
  - Work projects
  - Libraries