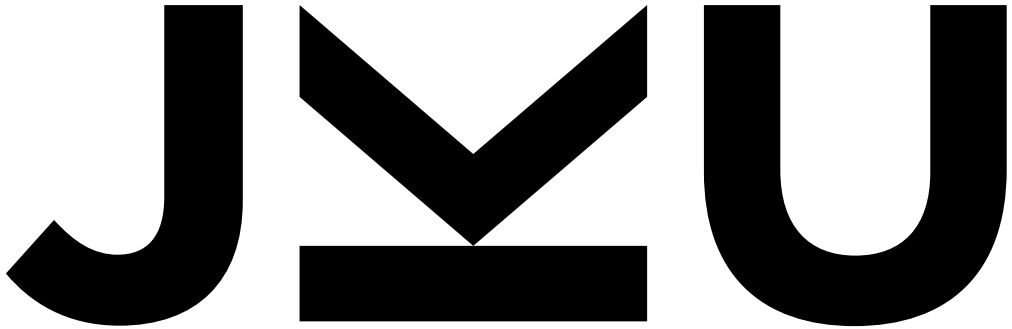
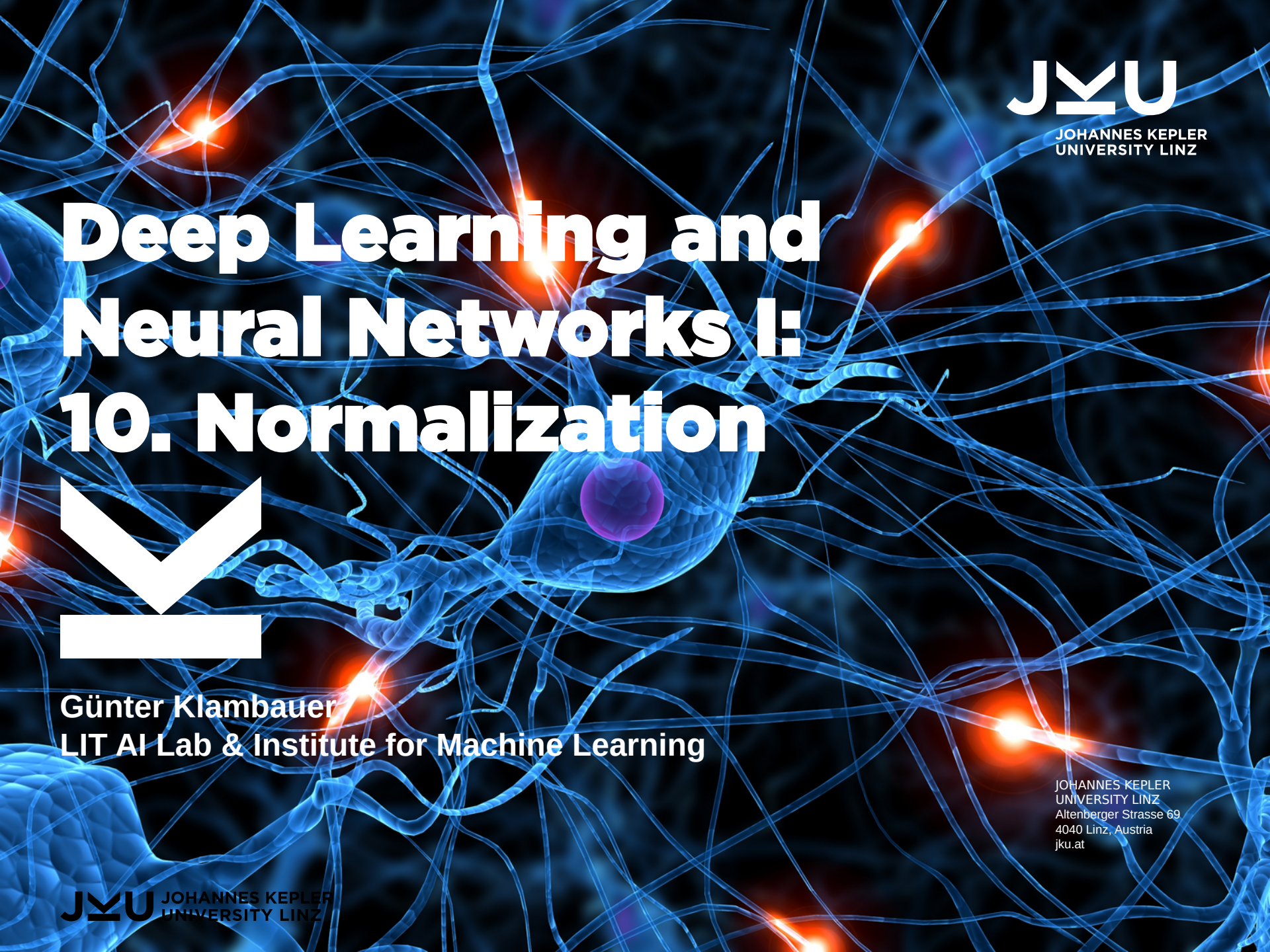


JOHANNES KEPLER
UNIVERSITY LINZ



JOHANNES KEPLER UNIVERSITY LINZ

# Deep Learning and Neural Networks I: 10. Normalization

Günter Klambauer
LIT AI Lab & Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Strasse 69
4040 Linz, Austria
jku.at

JOHANNES KEPLER UNIVERSITY LINZ

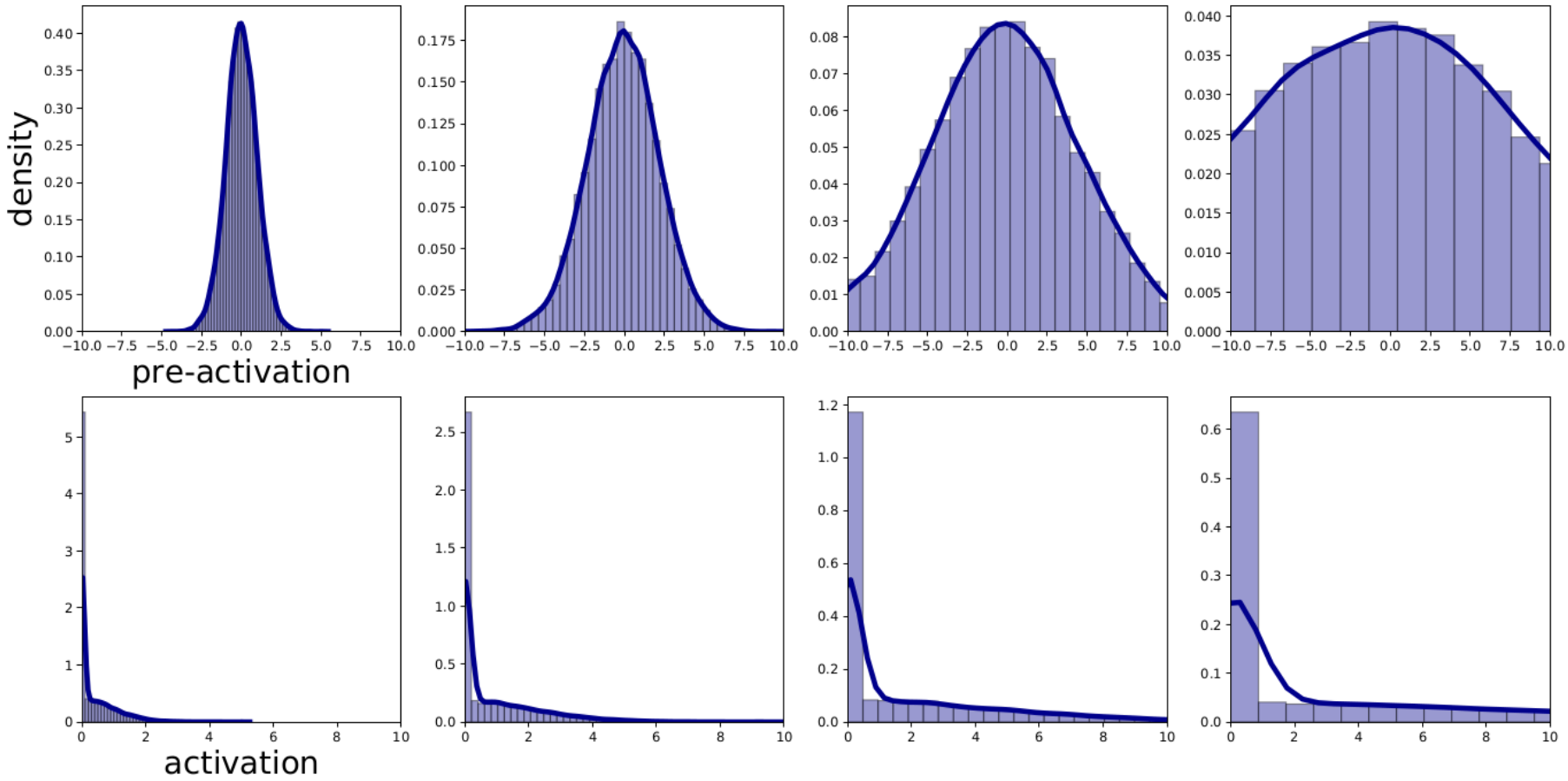
# Administrative

- Exam Friday Jan 29, 10:45, Online on Moodle and Zoom
  - 1.5h
  - Simple pocket calculator, sheet of paper and pen can be used
- Retry-exam: fixed, check KUSSS

# Overview

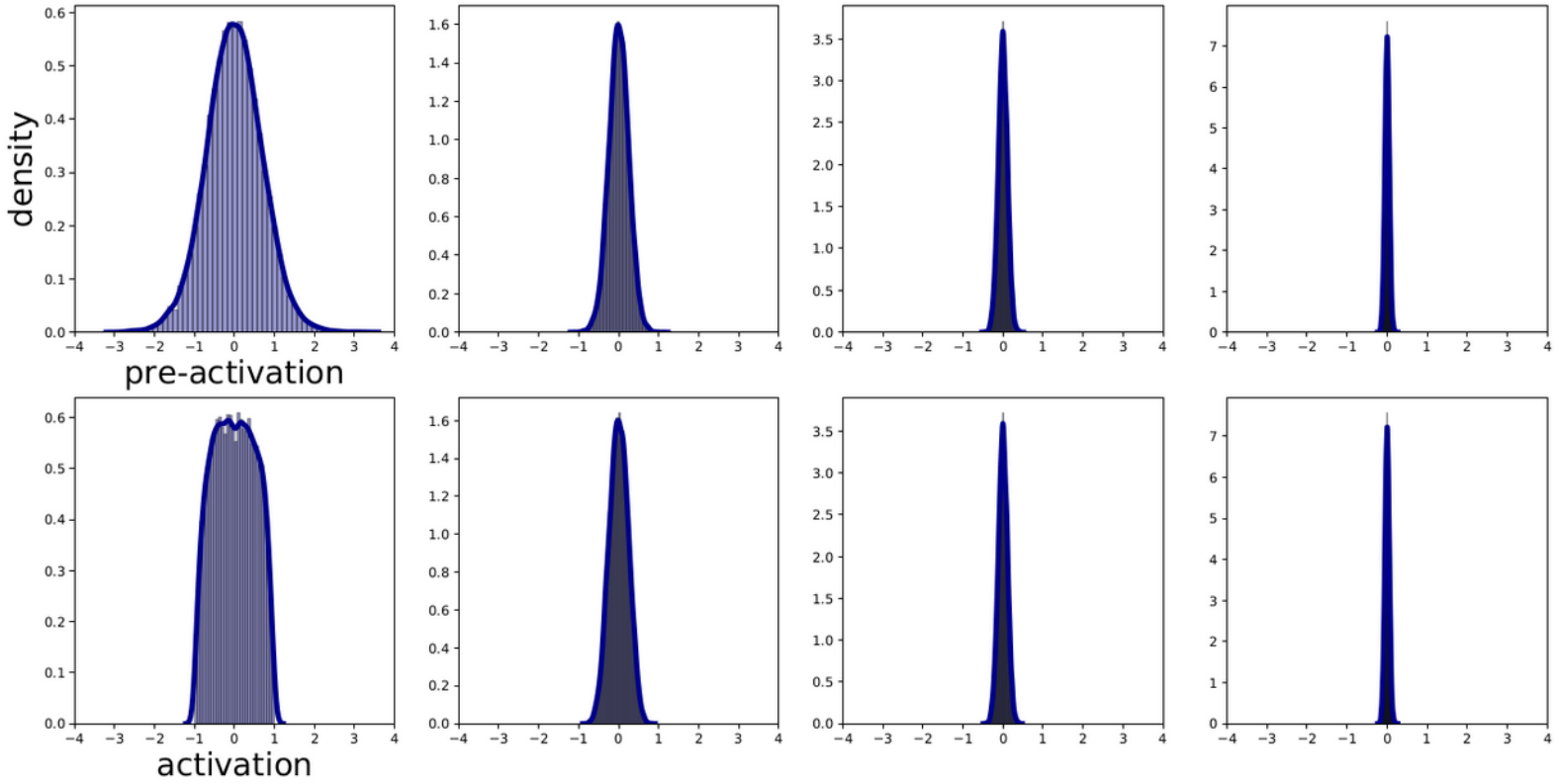
- 10.1 Batch normalization
- 10.2 Layer normalization
- 10.3 Weight normalization
- 10.4 Self-normalization
- 10.5 Invariance properties of normalization techniques

# Vanishing and exploding activations




Activation function: ReLU

# Vanishing and exploding activations




Activation function: Tanh

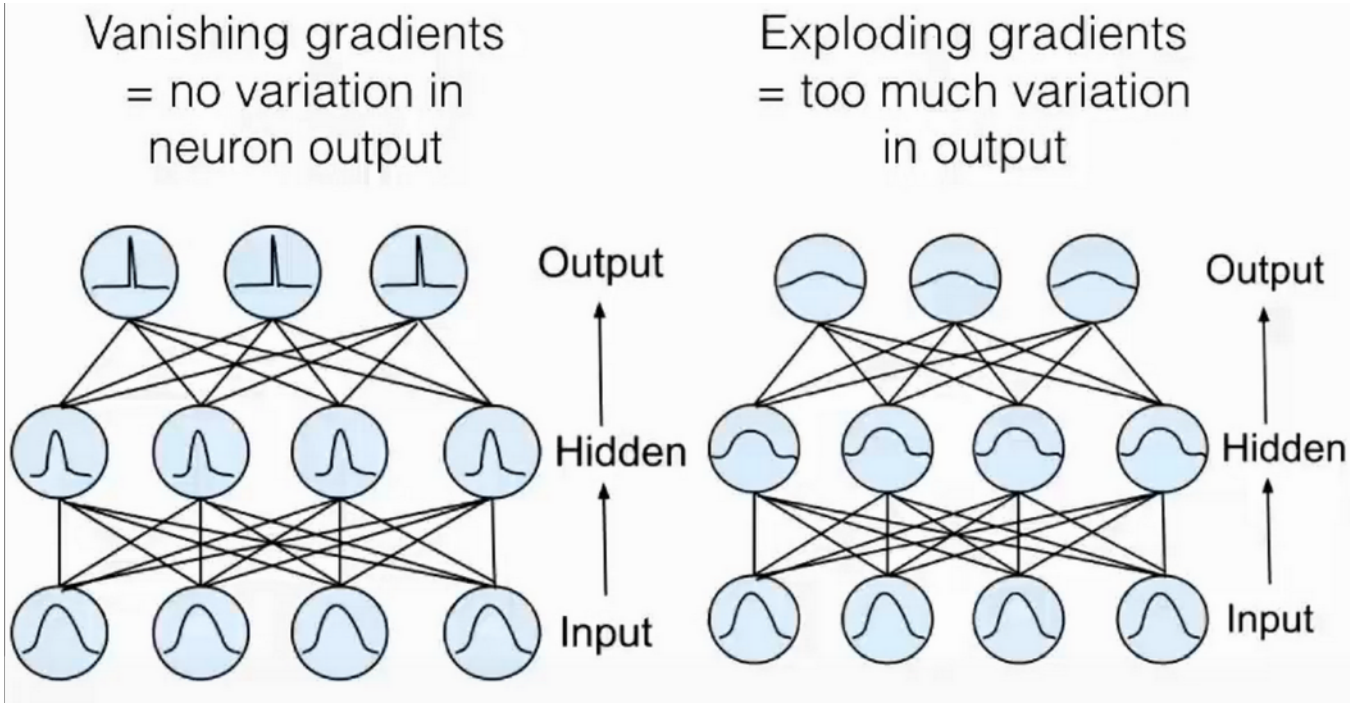
# Vanishing and exploding activations: Relation to vanishing gradient problem

- We remember from backpropagation:

$$\Delta w_{ij} = -\eta a_i \delta_j$$

- The size of the update depends on the activation, thus vanishing and exploding activations could be a problem.

# Vanishing and exploding activations: Relation to vanishing gradient problem

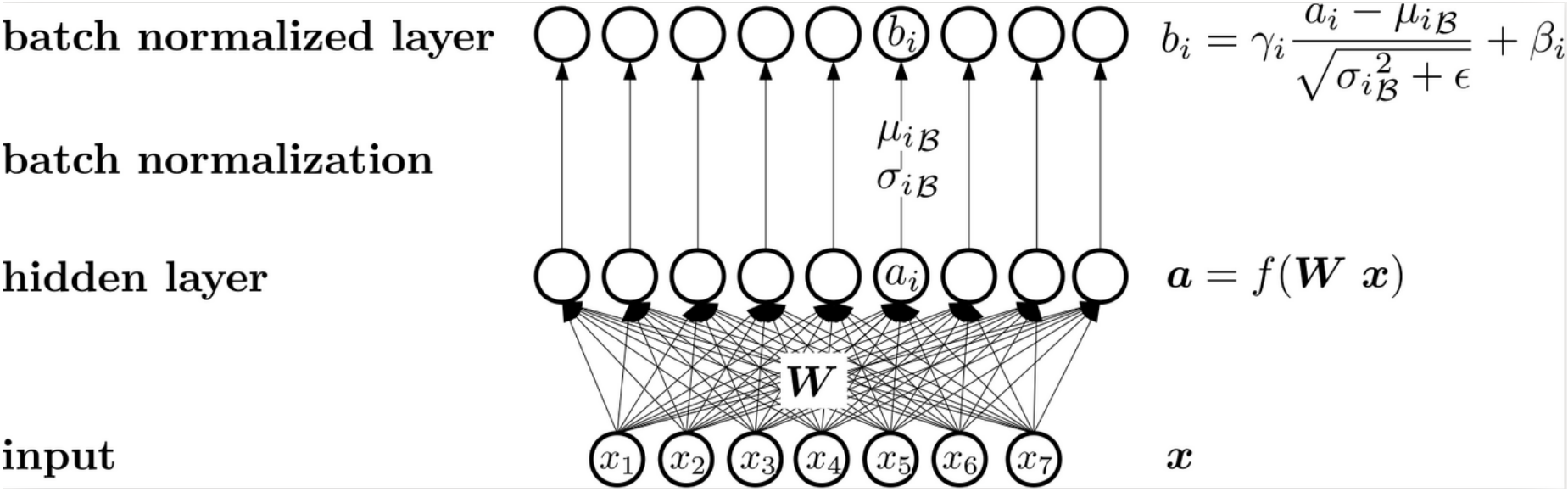



Salimans, T., & Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In Advances in Neural Information Processing Systems (pp. 901-909).

# Batch normalization

- Main idea: learn mean and variance of each neuron
- Problem: mean and variance change during learning
- Solution: learn mean and variance for each neuron per batch

# Batch normalization




- Each neuron “learns” a mean $\mu_i$ and a variance $\sigma_i^2$ for a batch
- The operation is differentiable wrt inputs

# Batch normalization (BN)

- Assume we have a neuron with activation $a$, then BN will transform:

$$\hat{a} = \frac{a - \mathbb{E}[a]}{\sqrt{\mathrm{Var}[a]}}$$

- Expected value and variance of training set
  - Estimated on a batch
- Batch normalization also introduces two learnable parameters $\beta$ and $\gamma$ for each neuron, which can scale and shift the normalized activation

$$b = \gamma\hat{a} + \beta.$$

# Batch normalization (BN)

- BN uses an estimation of $\mathbb{E}[a]$ and $\mathrm{Var}[a]$ based on a mini-batch $\mathcal{B}$ with $|\mathcal{B}| = B$.Index $i$ of neuron is dropped.
- Denoting the activations of our particular neurons in the mini-batch as $a^{(1)}, \ldots, a^{(B)}$ we formulate the *batch normalizing transformation*:

$$\mu_{\mathcal{B}} = \frac{1}{B} \sum_{n=1}^{B} a^{(n)}$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{B} \sum_{n=1}^{B} (a^{(n)} - \mu_{\mathcal{B}})^2$$

$$\hat{a}^{(\xi)} = \frac{a^{(\xi)} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad 1 \leq \xi \leq B$$

$$b^{(\xi)} = \gamma \hat{a}^{(\xi)} + \beta \qquad \epsilon \text{ constant}$$

# Batch normalization (BN)

**Forward pass:**

$$\mu_{\mathcal{B}} = \frac{1}{B}\sum_{n=1}^{B} a^{(n)}$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{B}\sum_{n=1}^{B} (a^{(n)} - \mu_{\mathcal{B}})^2$$

$$\hat{a}^{(\xi)} = \frac{a^{(\xi)} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad 1 \leq \xi \leq B$$

$$b^{(\xi)} = \gamma \hat{a}^{(\xi)} + \beta \qquad \epsilon \text{ constant}$$

- Backpropagation

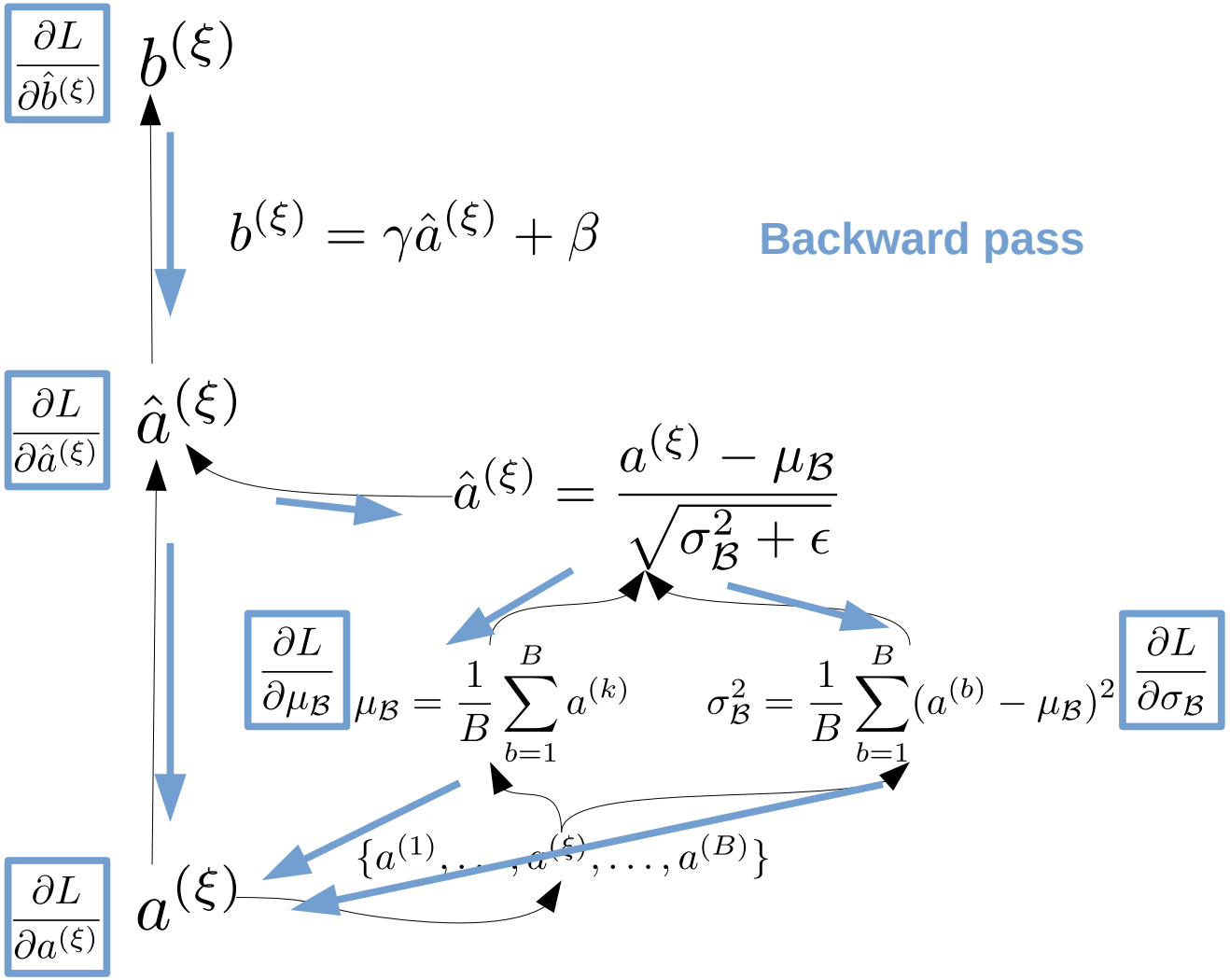
$$\frac{\partial L}{\partial \hat{a}^{(\xi)}} = \underbrace{\frac{\partial L}{\partial b^{(\xi)}}}_{:=\delta_{b(\xi)}} \gamma$$

$$\frac{\partial L}{\partial \sigma_{\mathcal{B}}^2} = \sum_{n=1}^{B} \frac{\partial L}{\partial \hat{a}^{(n)}} (a^{(n)} - \mu_{\mathcal{B}}) \frac{-1}{2} (\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial L}{\partial \mu_{\mathcal{B}}} = \left( \sum_{n=1}^{B} \frac{\partial L}{\partial \hat{a}^{(n)}} \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial L}{\partial \sigma_{\mathcal{B}}^2} \frac{\sum_{n=1}^{B} -2(a^{(n)} - \mu_{\mathcal{B}})}{B}$$

$$\frac{\partial L}{\partial a^{(\xi)}} = \frac{\partial L}{\partial \hat{a}^{(\xi)}} \underbrace{\frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}}_{\frac{\partial \hat{a}^{(\xi)}}{\partial a^{(\xi)}}} + \frac{\partial L}{\partial \sigma_{\mathcal{B}}^2} \underbrace{\frac{2(a^{(\xi)} - \mu_{\mathcal{B}})}{B}}_{\frac{\partial \sigma_{\mathcal{B}}^2}{\partial a^{(\xi)}}} + \frac{\partial L}{\partial \mu_{\mathcal{B}}} \underbrace{\frac{1}{B}}_{\frac{\partial \mu_{\mathcal{B}}}{\partial a^{(\xi)}}}$$

# Batch normalization (BN): backpropagation paths



# Batch normalization (BN)

**Forward pass:**

$$\mu_{\mathcal{B}} = \frac{1}{B}\sum_{n=1}^{B} a^{(n)}$$

$$\sigma^2_{\mathcal{B}} = \frac{1}{B}\sum_{n=1}^{B} (a^{(n)} - \mu_{\mathcal{B}})^2$$

$$\hat{a}^{(\xi)} = \frac{a^{(\xi)} - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad 1 \leq \xi \leq B$$

$$b^{(\xi)} = \gamma \hat{a}^{(\xi)} + \beta \qquad \epsilon \text{ constant}$$

$$\frac{\partial L}{\partial \gamma} = \sum_{n=1}^{B} \frac{\partial L}{\partial b^{(n)}} \hat{a}^{(n)}$$

$$\frac{\partial L}{\partial \beta} = \sum_{n=1}^{B} \frac{\partial L}{\partial b^{(n)}}$$

- During training only the parameters $\beta$ and $\gamma$ are updated with a gradient descent step.
  The expression $\frac{\partial L}{\partial a^{(k)}}$ is needed to provide delta-errors for the lower layer.
  The parameters $\mu_{\mathcal{B}}$ and $\sigma^2_{\mathcal{B}}$ are updated on-the-fly for each minibatch.

# Batch normalization (BN)

- We denote the whole batch normalizing transform as

$$b = \mathrm{BN}_{\beta,\gamma}(a).$$

- Properties: **Invariance** to weight scaling in linear and ReLU layers

  Assume we have a neuron $\mathrm{a}^{(n)} = \mathrm{ReLU}(\boldsymbol{w}^T \boldsymbol{x}^n)$ where $\boldsymbol{x}^n$ can either be the $n$-th input object or the activations for the $n$-th sample from the lower layer. We now find that the mean $\frac{1}{n}\sum_{n=1}^{N} a^{(n)}$ scales linearly with the scaling the weights by a constant $c > 0$ :

$$\frac{1}{n}\sum_{n=1}^{N} \mathrm{ReLU}(c\boldsymbol{w}^T \boldsymbol{x}^n) = c\frac{1}{n}\sum_{n=1}^{N} \mathrm{ReLU}(\boldsymbol{w}^T \boldsymbol{x}^n).$$

# Batch normalization (BN)

The second moment scales with $c^2$ :

$$\frac{1}{n}\sum_{n=1}^{N}\mathrm{ReLU}(c\boldsymbol{w}^T\boldsymbol{x}^n)^2 = c^2\frac{1}{n}\sum_{n=1}^{N}\mathrm{ReLU}(\boldsymbol{w}^T\boldsymbol{x}^n)^2.$$

Superscript $s$ : Quantities with scaled weights

$$\mu_{\mathcal{B}}^s = \frac{1}{B}\sum_{k=1}^{B}\mathrm{ReLU}(c\boldsymbol{w}^T\boldsymbol{x}^k) = c\ \mu_{\mathcal{B}}$$

$${\sigma_{\mathcal{B}}^s}^2 = \frac{1}{B}\sum_{k=1}^{B}(\mathrm{ReLU}(c\boldsymbol{w}^T\boldsymbol{x}^k) - \mu_{\mathcal{B}}^s)^2 = c^2\ \sigma_{\mathcal{B}}^2$$

$$\hat{a}^s = \frac{\mathrm{ReLU}(c\boldsymbol{w}^T\boldsymbol{x}) - c\mu_{\mathcal{B}}}{\sqrt{c^2\sigma_{\mathcal{B}}^2}} = \frac{\mathrm{ReLU}(\boldsymbol{w}^T\boldsymbol{x}) - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2}} = \hat{a}$$

# Invariance properties of normalization techniques (Ba et al., 2016)

| | weight matrix re-scaling | weight matrix re-centering | weight vector re-scaling | data set re-scaling | data set re-centering | single training case re-scaling |
|---|---|---|---|---|---|---|
| batch norm | invariant | no | invariant | invariant | invariant | no |
| weight norm | invariant | no | invariant | no | no | no |
| layer norm | invariant | invariant | no | invariant | no | invariant |
| self norm | no | no | no | no | no | no |

Table 10.1: Invariance properties of normalization methods according to (Ba et al., 2016). The invariance properties only hold if the normalization technique is applied on the pre-activations of full-connected layer (linear transformation).

# Layer normalization (LN)

- Main idea: All neurons in one layer share the same parametes $\mu$ and $\sigma$.
- Assume that the inpunts to the layer normalizing layer are pre-activations $s$, a vector of length $J$.
  *Layer normalization:*

$$\mu = \frac{1}{J}\sum_{j=1}^{J} s_j$$

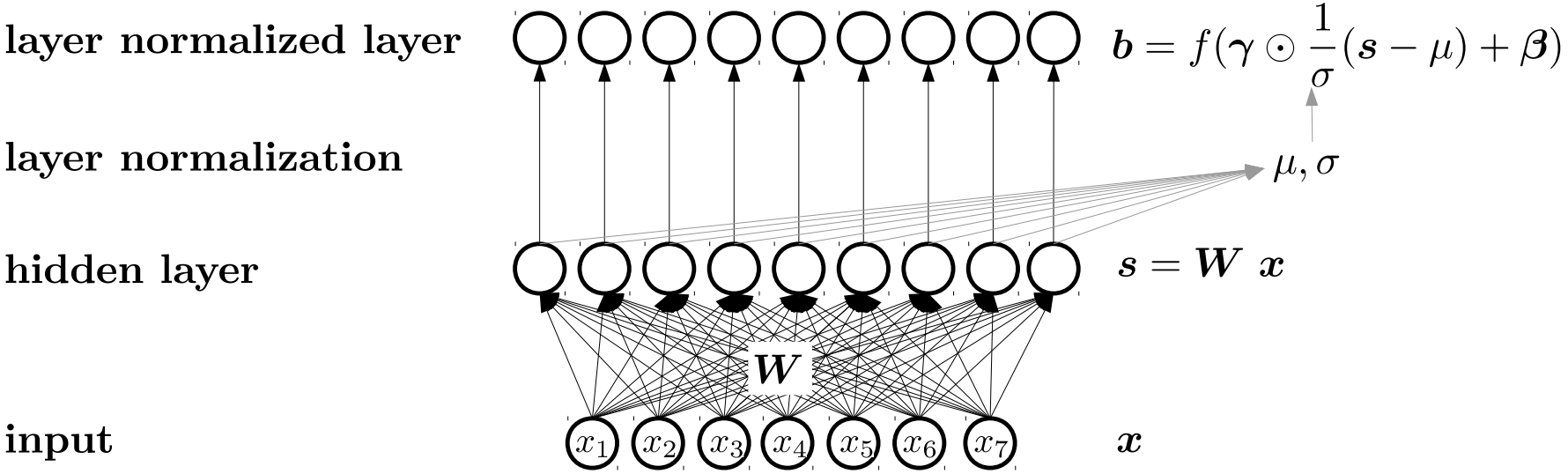
$$\sigma^2 = \frac{1}{J}\sum_{j=1}^{J}(s_j - \mu)^2$$

$$\boldsymbol{b} = f(\boldsymbol{\gamma} \odot \frac{1}{\sigma}(\boldsymbol{s} - \mu) + \boldsymbol{\beta})$$

$\mu, \sigma$ :
Mean and standard deviation of the respective layer

$\boldsymbol{\gamma}, \boldsymbol{\beta}$ :
Learnable parametes

# Layer normalization



# Weight normalization (WN)

- Assume that a neuron in the network has the activation $a = \mathrm{ReLU}(\boldsymbol{w}^T \boldsymbol{x})$.

- Main idea: Reparameterize the weights in terms of a new weight vector $\boldsymbol{v}$ and a parameter $g$, which is called *weight normalization:*

$$\boldsymbol{w} = \frac{g}{\|\boldsymbol{v}\|}\boldsymbol{v},$$

*where the vector $\boldsymbol{v}$ has the same dimension as $\boldsymbol{w}$ and is divided by its Euclidean norm.*

# Weight normalization

- *We further assume that there is a loss function $L(\boldsymbol{y}, g(\boldsymbol{x}, \boldsymbol{w}))$ which depends on $\boldsymbol{w}$, such that we can have a gradient with respect to the parameters $\boldsymbol{w}$: $\nabla_{\boldsymbol{w}} L$, then:*

$$\nabla_g L = \frac{(\nabla_{\boldsymbol{w}} L) \cdot \boldsymbol{v}}{\|\boldsymbol{v}\|}$$

$$\nabla_{v_i} L = \frac{g}{\|\boldsymbol{v}\|} \left(1 - \frac{v_i^2}{\|\boldsymbol{v}\|^2}\right) \nabla_{w_i} L,$$

*where $\nabla_{\boldsymbol{w}} L$ is a row vector and we have used the quotient rule for the second term.*

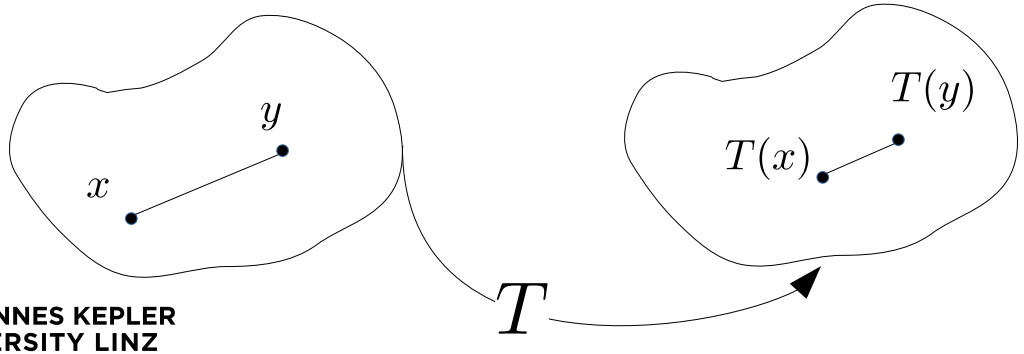
# Math background for SNNs: Banach fixed point theorem

**Definition 1** *Let $(X, d)$ be a complete metric space. Then a mapping $T : X \mapsto X$ is called* contraction *mapping on $X$ if there exists $q \in [0, 1)$ such that*

$$d(T(x), T(y)) \leq q \; d(x, y)$$

*for all $x, y \in X$.*

**Theorem .1 (Banach fixed point theorem)** *Let $(X, d)$ be a non-empty complete metric space with a contraction mapping $T : X \mapsto X$. Then $T$ admits an unique fixed-point $x^*$ in $X$, i.e. $T(x^*) = x^*$. Furthermore, $x^*$ can be found in the following way: start with an arbitrary $x^{(0)} \in X$ and then follow $x^{(t+1)} = T(x^{(t)})$ for $t \geq 0$. Then $x^{(t)} \to x^*$.*



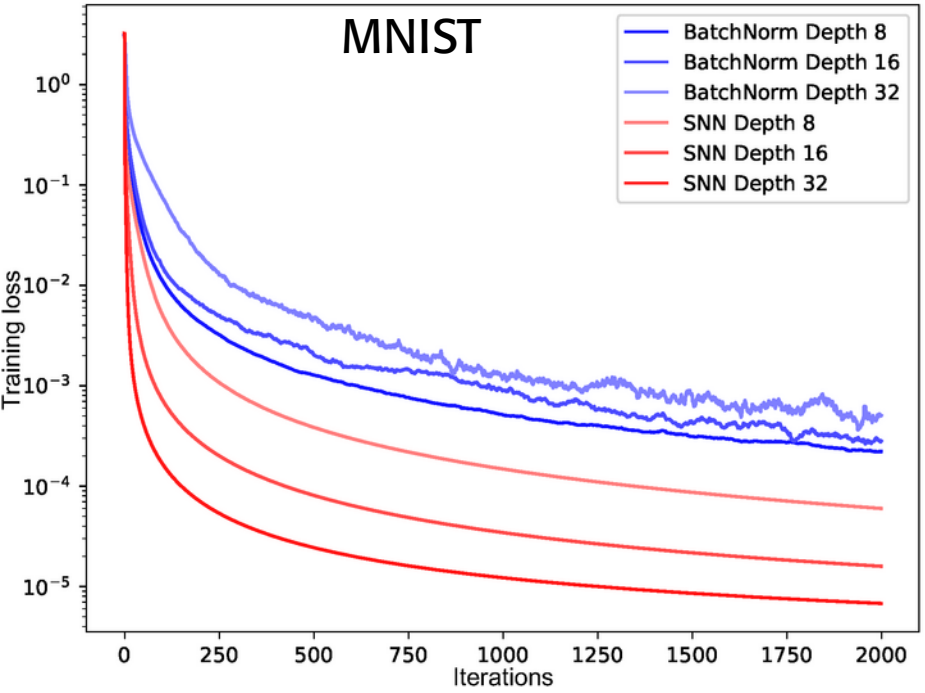

Jacobian of map smaller 1: contraction map

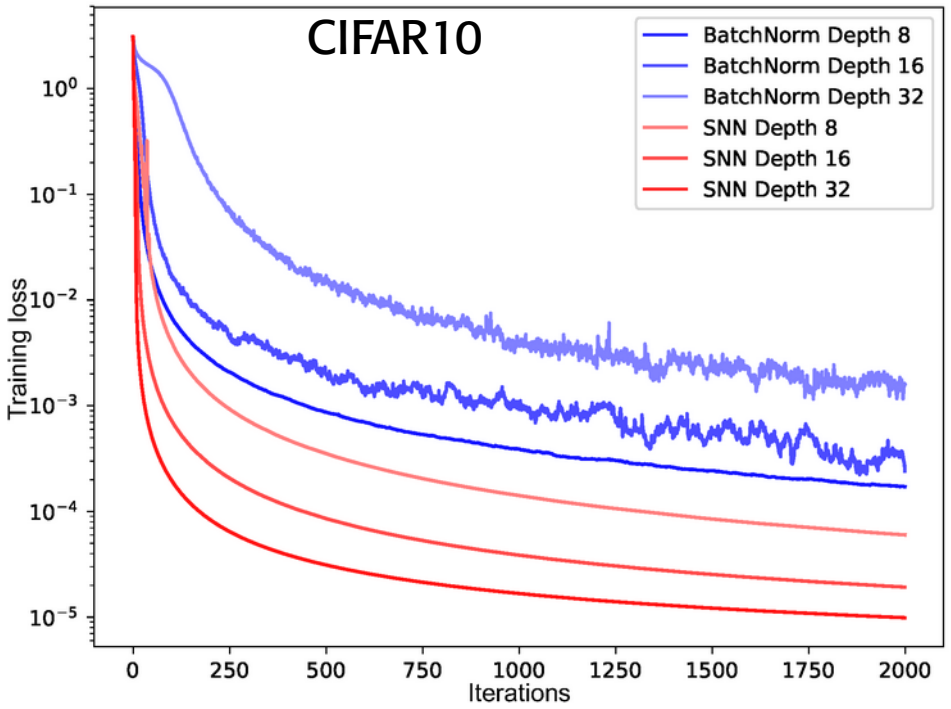
$\|J_T\| < 1$

# Explicit Normalization methods

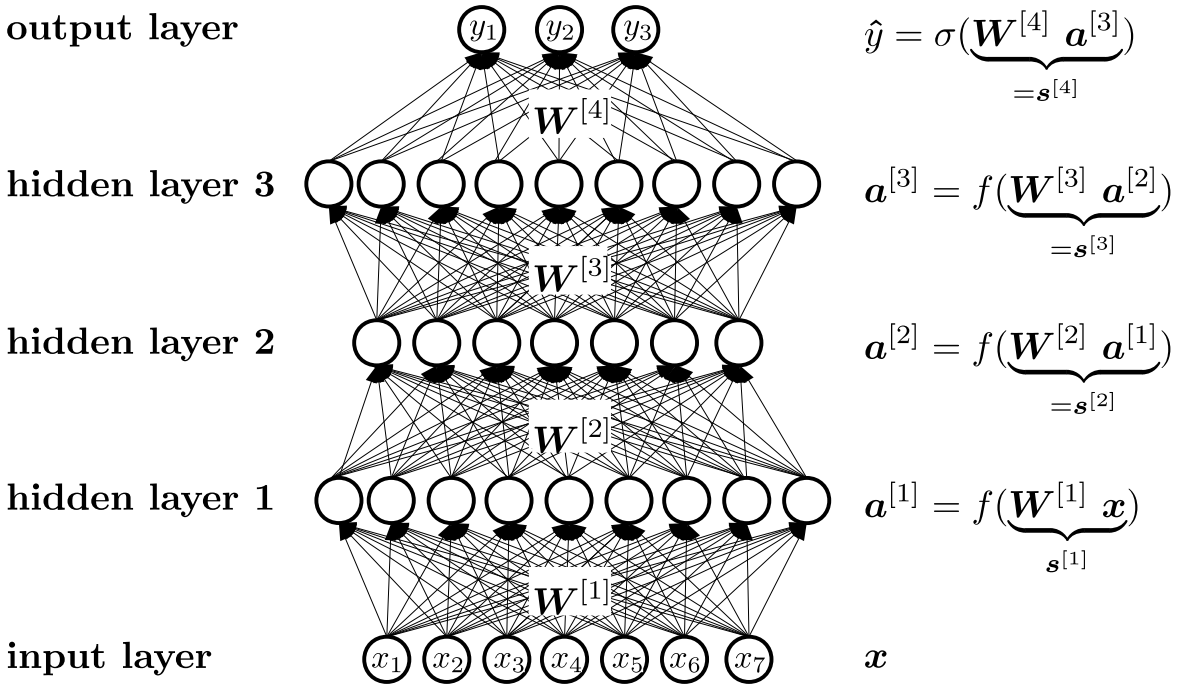
- As discussed:
  - **Batch normalization:** keep distributions of activations per batch similar (zero mean, unit variance)
  - **Layer normalization:** keep distributions of activations per batch similar (zero mean, unit variance)
  - **Weight normalization:** normalize weights to keep activations close to zero mean and unit variance
- New idea: equip a neural network with the ability to automatically normalize itself

# Problems with external normalization



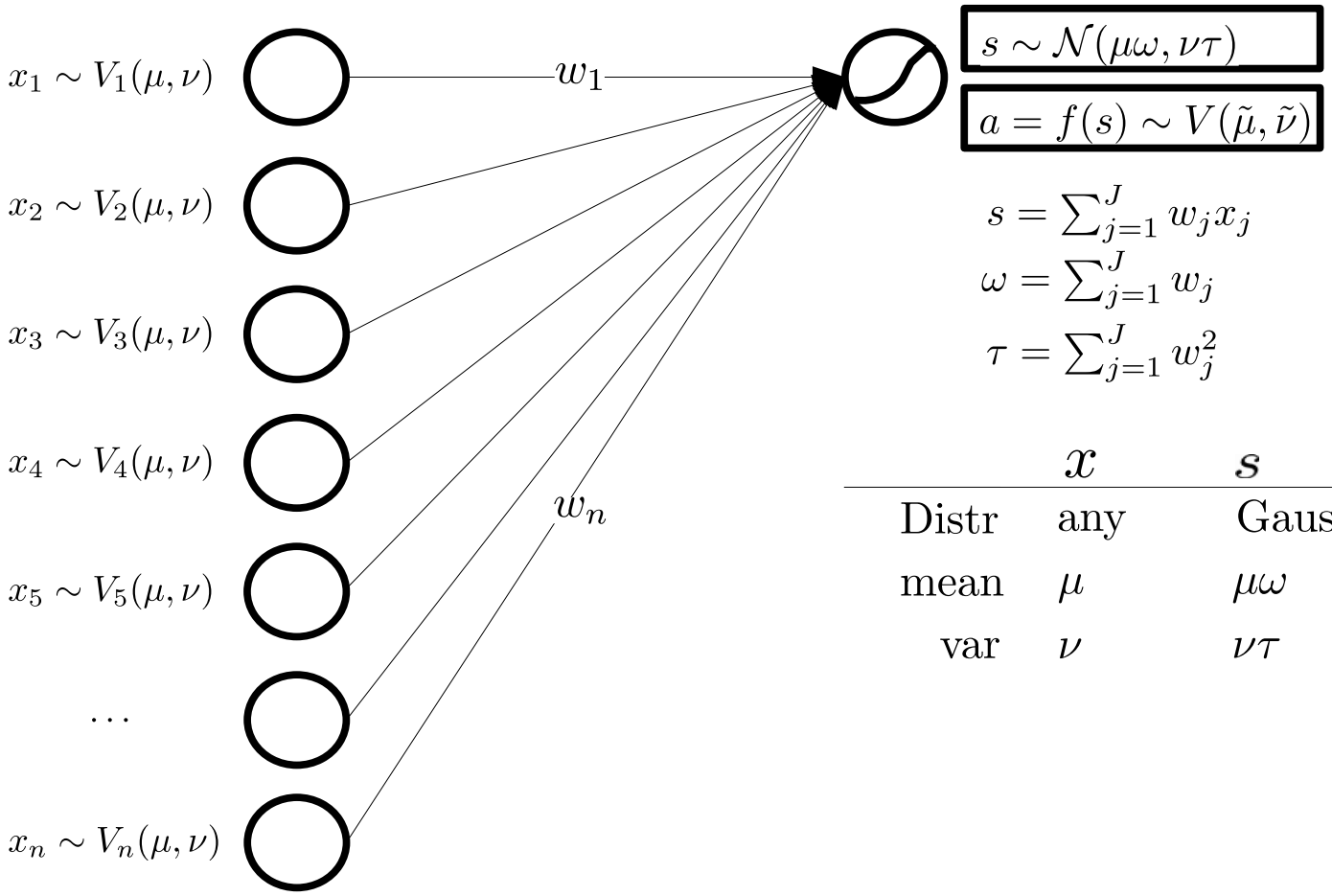
# Forward pass of a deep neural network




- Mapping from one layer to the next
  - Linear mapping
  - Non-linear activation
- Large matrices
- Many layers

- Network inputs: $\boldsymbol{s} = \boldsymbol{W}\boldsymbol{x}$ $s = \boldsymbol{w}^T\boldsymbol{x}$ Activations: $\boldsymbol{a} = f(\boldsymbol{s})$
- Activations of lower layer: $\mu := \mathbb{E}(x_j)$ $\nu := \mathrm{Var}(x_j)$

Network weights: $\omega := \sum_{j=1}^{J} w_j$ $\tau := \sum_{j=1}^{J} w_j^2$

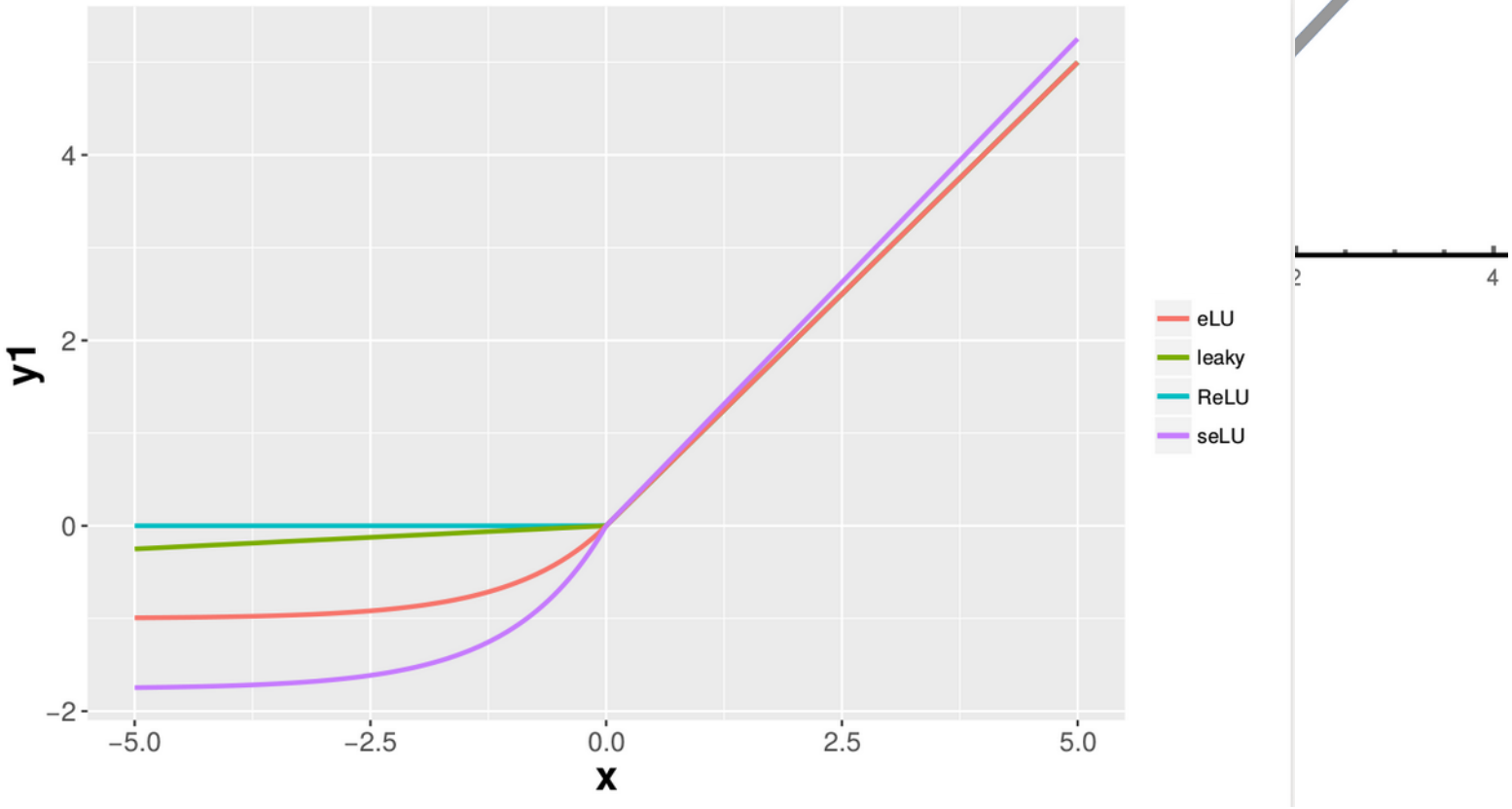



| | $x$ | $s$ | $a$ |
|---|---|---|---|
| Distr | any | Gauss | any |
| mean | $\mu$ | $\mu\omega$ | $\tilde{\mu}$ |
| var | $\nu$ | $\nu\tau$ | $\tilde{\nu}$ |

# Activation function: scaled exponential linear unit (SELU)

- SELU:
  - -

$$\text{selu}(x)$$



# Self-normalizing networks: Use an activation function “SELU” to keep mean and variance

$$g : \begin{pmatrix} \mu \\ \nu \end{pmatrix} \mapsto \begin{pmatrix} \tilde{\mu} \\ \tilde{\nu} \end{pmatrix} : \qquad \tilde{\mu}(\mu, \omega, \nu, \tau) \;=\; \int_{-\infty}^{\infty} \mathrm{selu}(z)\; p(z; \mu\omega, \sqrt{\nu\tau})\; \mathrm{d}z$$

$$\tilde{\nu}(\mu, \omega, \nu, \tau) \;=\; \int_{-\infty}^{\infty} \mathrm{selu}(z)^2\; p(z; \mu\omega, \sqrt{\nu\tau})\; \mathrm{d}z \;-\; (\tilde{\mu})^2 \;.$$

$$\tilde{\mu}(\mu, \omega, \nu, \tau, \lambda, \alpha) \;=\; \frac{1}{2}\lambda \left( (\mu\omega) \left( \frac{\mu\omega}{\sqrt{2}\sqrt{\nu\tau}} \right) + \alpha e^{\mu\omega + \frac{\nu\tau}{2}} \left( \frac{\mu\omega + \nu\tau}{\sqrt{2}\sqrt{\nu\tau}} \right) - \alpha \left( \frac{\mu\omega}{\sqrt{2}\sqrt{\nu\tau}} \right) + \sqrt{\frac{2}{\pi}} \sqrt{\nu\tau} e^{-\frac{(\mu\omega)^2}{2(\nu\tau)}} + \mu\omega \right)$$

$$\tilde{\nu}(\mu, \omega, \nu, \tau, \lambda, \alpha) \;=\; \frac{1}{2}\lambda^2 \left( ((\mu\omega)^2 + \nu\tau) \left( \left( \frac{\mu\omega}{\sqrt{2}\sqrt{\nu\tau}} \right) + 1 \right) + \alpha^2 \left( -2 e^{\mu\omega + \frac{\nu\tau}{2}} \left( \frac{\mu\omega + \nu\tau}{\sqrt{2}\sqrt{\nu\tau}} \right) + e^{2(\mu\omega + \nu\tau)} \left( \frac{\mu\omega + 2\nu\tau}{\sqrt{2}\sqrt{\nu\tau}} \right) + \left( \frac{\mu\omega}{\sqrt{2}\sqrt{\nu\tau}} \right) \right) + \sqrt{\frac{2}{\pi}} (\mu\omega) \sqrt{\nu\tau} e^{-\frac{(\mu\omega)^2}{2(\nu\tau)}} \right) - {\mu^{\mathrm{new}}}^2$$

# Solving fixed point equations for SELU parameters

$$\tilde{\mu}(\mu,\omega,\nu,\tau,\lambda,\alpha) = \frac{1}{2}\lambda\left((\mu\omega)\left(\frac{\mu\omega}{\sqrt{2}\sqrt{\nu\tau}}\right)+\alpha e^{\mu\omega+\frac{\nu\tau}{2}}\left(\frac{\mu\omega+\nu\tau}{\sqrt{2}\sqrt{\nu\tau}}\right)-\alpha\left(\frac{\mu\omega}{\sqrt{2}\sqrt{\nu\tau}}\right)+\sqrt{\frac{2}{\pi}}\sqrt{\nu\tau}e^{-\frac{(\mu\omega)^2}{2(\nu\tau)}}+\mu\omega\right)$$

$$\tilde{\nu}(\mu,\omega,\nu,\tau,\lambda,\alpha) = \frac{1}{2}\lambda^2\left(((\mu\omega)^2+\nu\tau)\left(\left(\frac{\mu\omega}{\sqrt{2}\sqrt{\nu\tau}}\right)+1\right)+\alpha^2\left(-2e^{\mu\omega+\frac{\nu\tau}{2}}\left(\frac{\mu\omega+\nu\tau}{\sqrt{2}\sqrt{\nu\tau}}\right)+e^{2(\mu\omega+\nu\tau)}\left(\frac{\mu\omega+2\nu\tau}{\sqrt{2}\sqrt{\nu\tau}}\right)+\left(\frac{\mu\omega}{\sqrt{2}\sqrt{\nu\tau}}\right)\right)+\sqrt{\frac{2}{\pi}}(\mu\omega)\sqrt{\nu\tau}e^{-\frac{(\mu\omega)^2}{2(\nu\tau)}}\right)-\mu^{\text{new}^2}$$

- Fixed point equations:

$$\tilde{\mu}=\mu=0$$
$$\tilde{\nu}=\nu=1$$

- Solving for alpha and lambda:

$$\alpha_{01}=-\frac{\sqrt{\frac{2}{\pi}}}{e^{\frac{1}{2}}\operatorname{erfc}\left(\frac{1}{\sqrt{2}}\right)-1}\approx 1.67$$

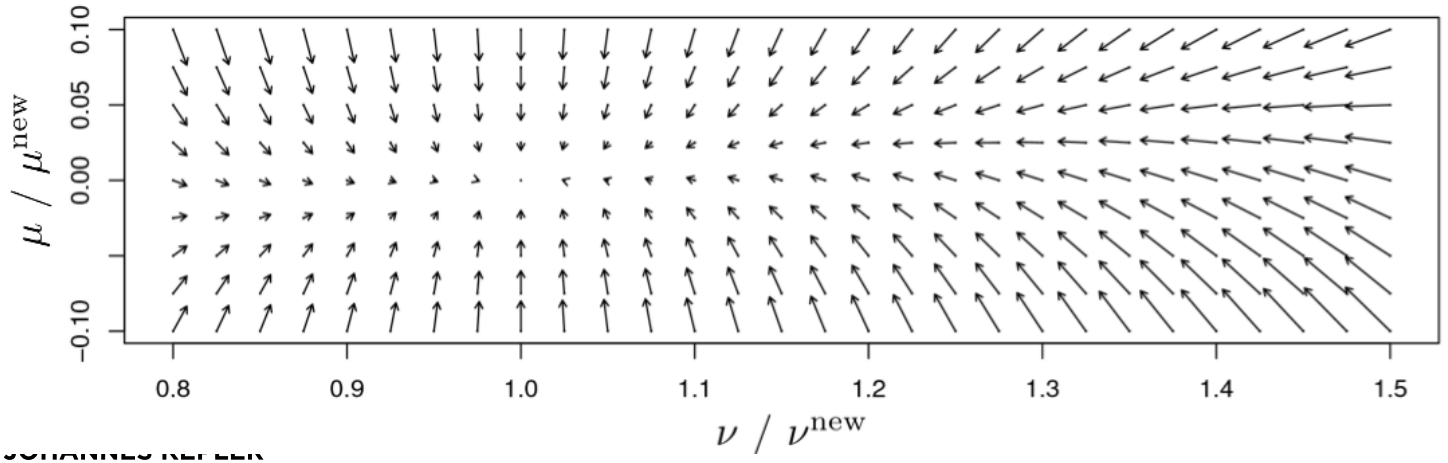
$$\lambda_{01}=\left(1-\sqrt{e}\operatorname{erfc}\left(\frac{1}{\sqrt{2}}\right)\right)\sqrt{\frac{2\pi}{e\pi\operatorname{erfc}\left(\frac{1}{\sqrt{2}}\right)^2-2\sqrt{e}(2+\pi)\operatorname{erfc}\left(\frac{1}{\sqrt{2}}\right)+2e^2\operatorname{erfc}\left(\sqrt{2}\right)+\pi+2}}\approx 1.05$$

# Stable and attracting fixed point (0,1) for normalized weights

- Jacobian of mapping g at fixed point (0,1):

$$\mathcal{J}(\mu,\nu) = \begin{pmatrix} \partial\frac{\mu(\mu,\nu)}{\partial\mu} & \partial\frac{\mu(\mu,\nu)}{\partial\nu} \\ \partial\frac{\nu(\mu,\nu)}{\partial\mu} & \partial\frac{\nu(\mu,\nu)}{\partial\nu} \end{pmatrix}, \qquad \mathcal{J}(0,1) = \begin{pmatrix} 0.0 & 0.088834 \\ 0.0 & 0.782648 \end{pmatrix}.$$

- spectral norm of Jacobian is smaller than one
- mapping $g$ is a contraction mapping
- (0,1) is a stable fixed point



# Theorem 1

$$g : \begin{pmatrix} \mu \\ \nu \end{pmatrix} \longrightarrow \begin{pmatrix} \hat{\mu} \\ \hat{\nu} \end{pmatrix}$$

**Theorem 1** (Stable and Attracting Fixed Points)**.** *We assume* $\alpha = \alpha_{01}$ *and* $\lambda = \lambda_{01}$*. We restrict the range of the variables to the following intervals* $\mu \in [-0.1, 0.1]$*,* $\omega \in [-0.1, 0.1]$*,* $\nu \in [0.8, 1.5]$*, and* $\tau \in [0.95, 1.1]$*, that define the functions' domain* $\Omega$*. For* $\omega = 0$ *and* $\tau = 1$*, the mapping* $g$ *has the stable fixed point* $(\mu, \nu) = (0, 1)$*, whereas for other* $\omega$ *and* $\tau$ *the mapping* $g$ *has a stable and attracting fixed point depending on* $(\omega, \tau)$ *in the* $(\mu, \nu)$*-domain:* $\mu \in [-0.03106, 0.06773]$ *and* $\nu \in [0.80009, 1.48617]$*. All points within the* $(\mu, \nu)$*-domain converge when iteratively applying the mapping* $g$ *to this fixed point.*

# Stable and attracting fixed points for non-normalized weights

- Proof sketch
  - Mapping from mean and variance of one layer to the next layer

$$g : \begin{pmatrix} \mu \\ \nu \end{pmatrix} \longrightarrow \begin{pmatrix} \hat{\mu} \\ \hat{\nu} \end{pmatrix}$$

  - $g$ is a contraction mapping
    - Matrix-Norm of Jacobian mapping (largest singular value) is below 1
    - Computer-assisted proof: bounded derivative
  - Mapping stays in domain $\Omega$
  - Banach's fixed point theorem can be applied to show that a unique fixed point exists that is attained
- Further theorems and proofs
  - Large variance: variance is contracting
  - Small variance: variance is expanding

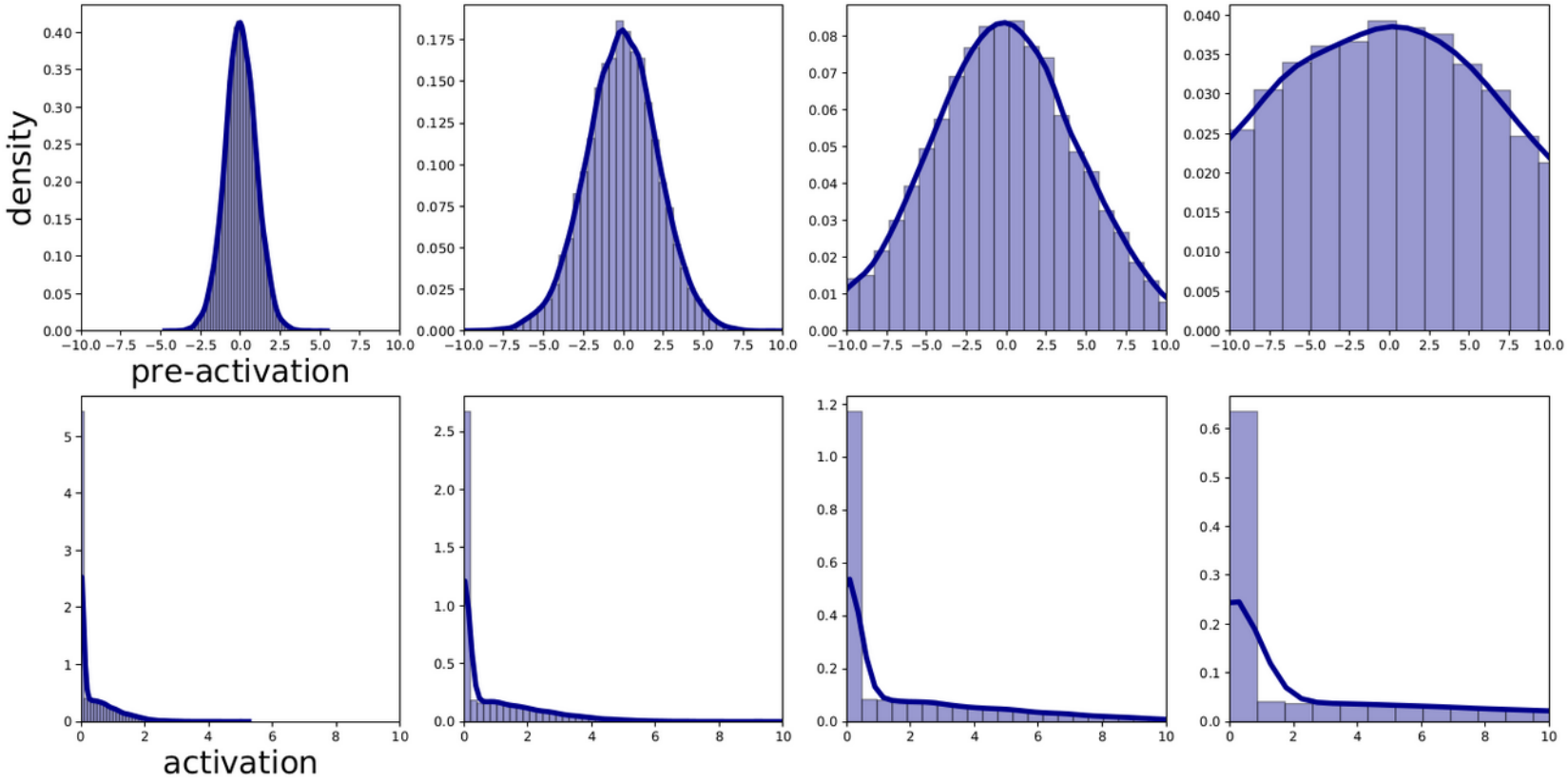
# Theorems 2 and 3

$$\tilde{\nu}(\mu,\omega,\nu,\tau,\lambda,\alpha) = \frac{1}{2}\lambda^2\left(((\mu\omega)^2+\nu\tau)\left(\left(\frac{\mu\omega}{\sqrt{2}\sqrt{\nu\tau}}\right)+1\right)+\alpha^2\left(-2e^{\mu\omega+\frac{\nu\tau}{2}}\left(\frac{\mu\omega+\nu\tau}{\sqrt{2}\sqrt{\nu\tau}}\right)+e^{2(\mu\omega+\nu\tau)}\left(\frac{\mu\omega+2\nu\tau}{\sqrt{2}\sqrt{\nu\tau}}\right)+\left(\frac{\mu\omega}{\sqrt{2}\sqrt{\nu\tau}}\right)\right)+\sqrt{\frac{2}{\pi}}(\mu\omega)\sqrt{\nu\tau}e^{-\frac{(\mu\omega)^2}{2(\nu\tau)}}\right)-\mu^{\text{new}^2}$$

**Theorem 2.** *For* $\lambda=\lambda_{01}$, $\alpha=\alpha_{01}$ *and the domain* $\Omega^+$*:* $-1\le\mu\le 1$, $-0.1\le\omega\le 0.1$, $3\le\nu\le 16$, *and* $0.8\le\tau\le 1.25$, *we have for the mapping of the variance* $\tilde{\nu}(\mu,\omega,\nu,\tau,\lambda,\alpha)$ *given in* $g$*:* $\tilde{\nu}(\mu,\omega,\nu,\tau,\lambda_{01},\alpha_{01})<\nu$.

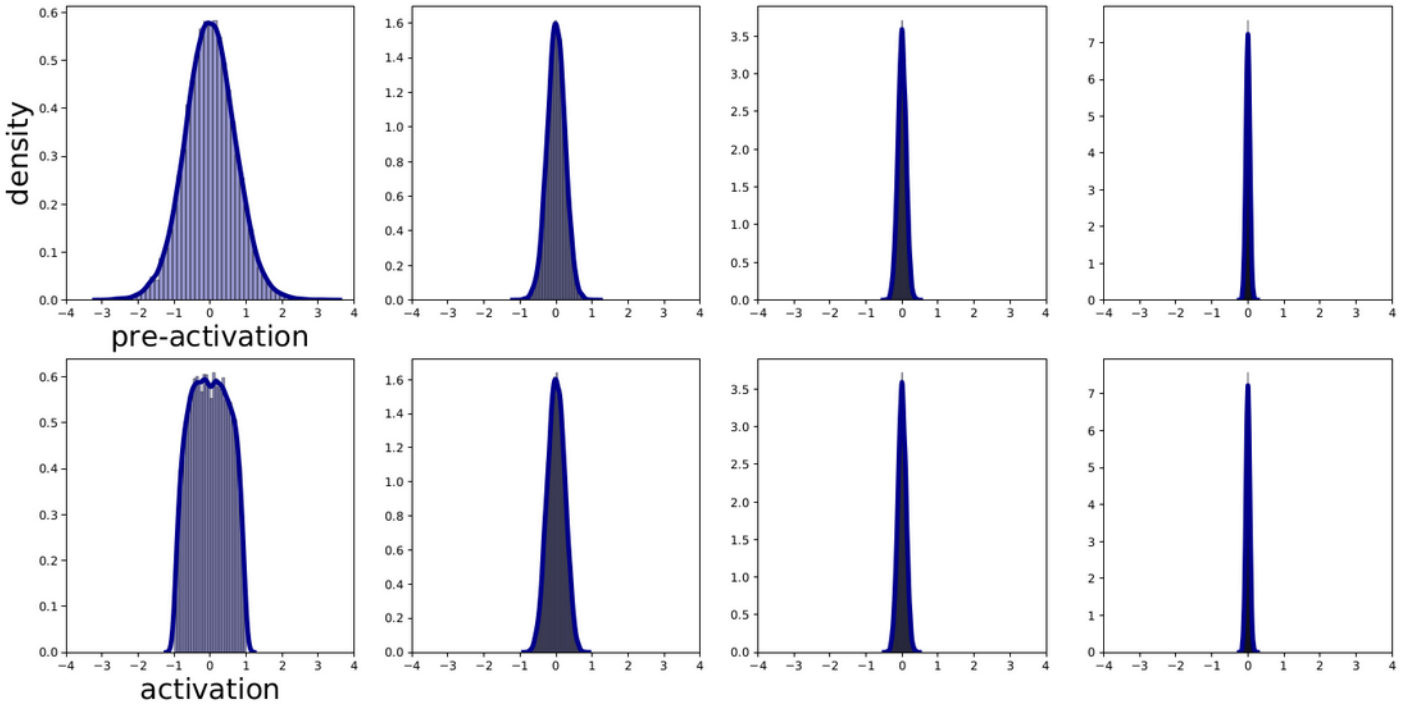
**Theorem 3.** *We consider* $\lambda=\lambda_{01}$, $\alpha=\alpha_{01}$ *and the domain* $\Omega^-$*:* $-0.1\le\mu\le 0.1$, *and* $-0.1\le\omega\le 0.1$. *For the domain* $0.02\le\nu\le 0.16$ *and* $0.8\le\tau\le 1.25$ *as well as for the domain* $0.02\le\nu\le 0.24$ *and* $0.9\le\tau\le 1.25$, *the mapping of the variance* $\tilde{\nu}(\mu,\omega,\nu,\tau,\lambda,\alpha)$ *given in* $g$ *increases:* $\tilde{\nu}(\mu,\omega,\nu,\tau,\lambda_{01},\alpha_{01})>\nu$.

# Reminder:
# Vanishing and exploding activations



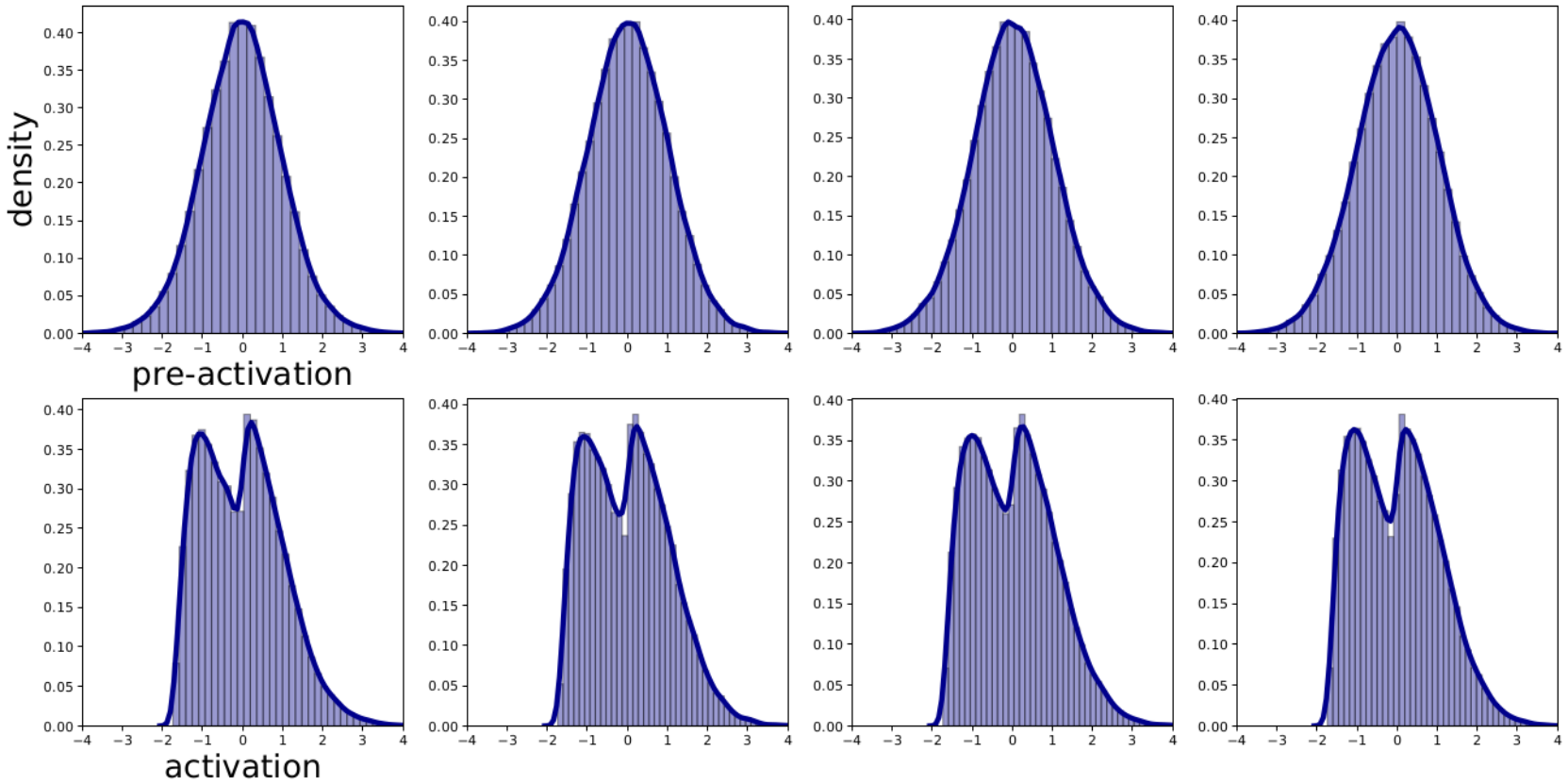

Activation function: ReLU

# Reminder:
# Vanishing and exploding activations




Activation function: Tanh

# “Vanishing and exploding activations”: SNNs



# Experiments

- 121 UCI machine learning repository data sets
- Tox21 data set
- HTRU2 data set
- Compared methods: Hyperparams adjusted on validation set
  - Unnormalized ReLU networks
  - Batch normalized networks
  - Weight normalized networks
  - Layer normalized networks
  - Highway networks
  - Residual networks
  - Self-normalizing networks

# UCI data set collection

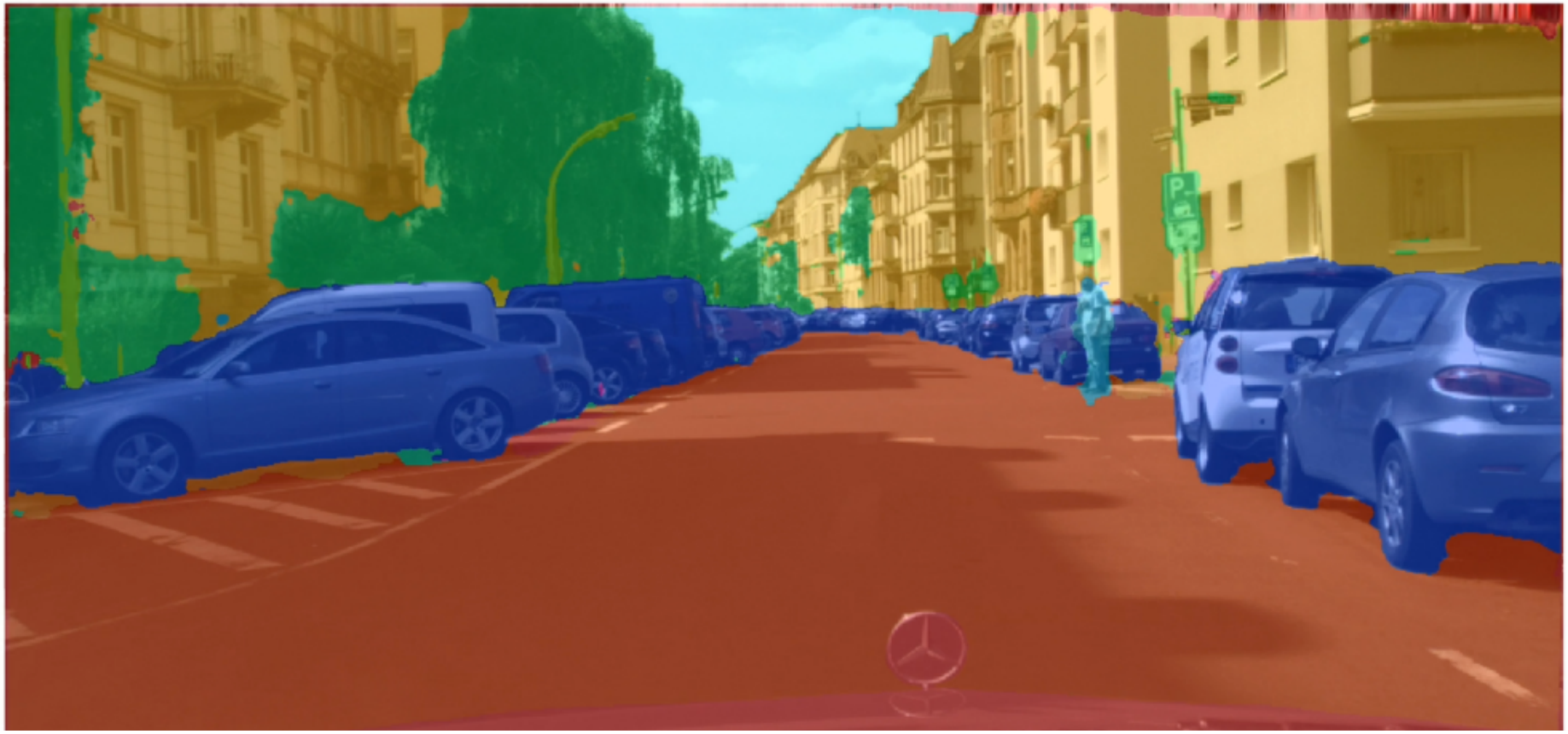
- 121 UCI machine learning data sets
- ~170 methods (SVMs, RF, GBMs) trained on

| FNN method comparison | | | ML method comparison | | |
|---|---|---|---|---|---|
| Method | avg. rank diff. | $p$-value | Method | avg. rank diff. | $p$-value |
| SNN | -0.756 | | SNN | -6.7 | |
| MSRAinit | -0.240* | 2.7e-02 | SVM | -6.4 | 5.8e-01 |
| LayerNorm | -0.198* | 1.5e-02 | RandomForest | -5.9 | 2.1e-01 |
| Highway | 0.021* | 1.9e-03 | MSRAinit | -5.4* | 4.5e-03 |
| ResNet | 0.273* | 5.4e-04 | LayerNorm | -5.3 | 7.1e-02 |
| WeightNorm | 0.397* | 7.8e-07 | Highway | -4.6* | 1.7e-03 |
| BatchNorm | 0.504* | 3.5e-06 | ... | ... | ... |

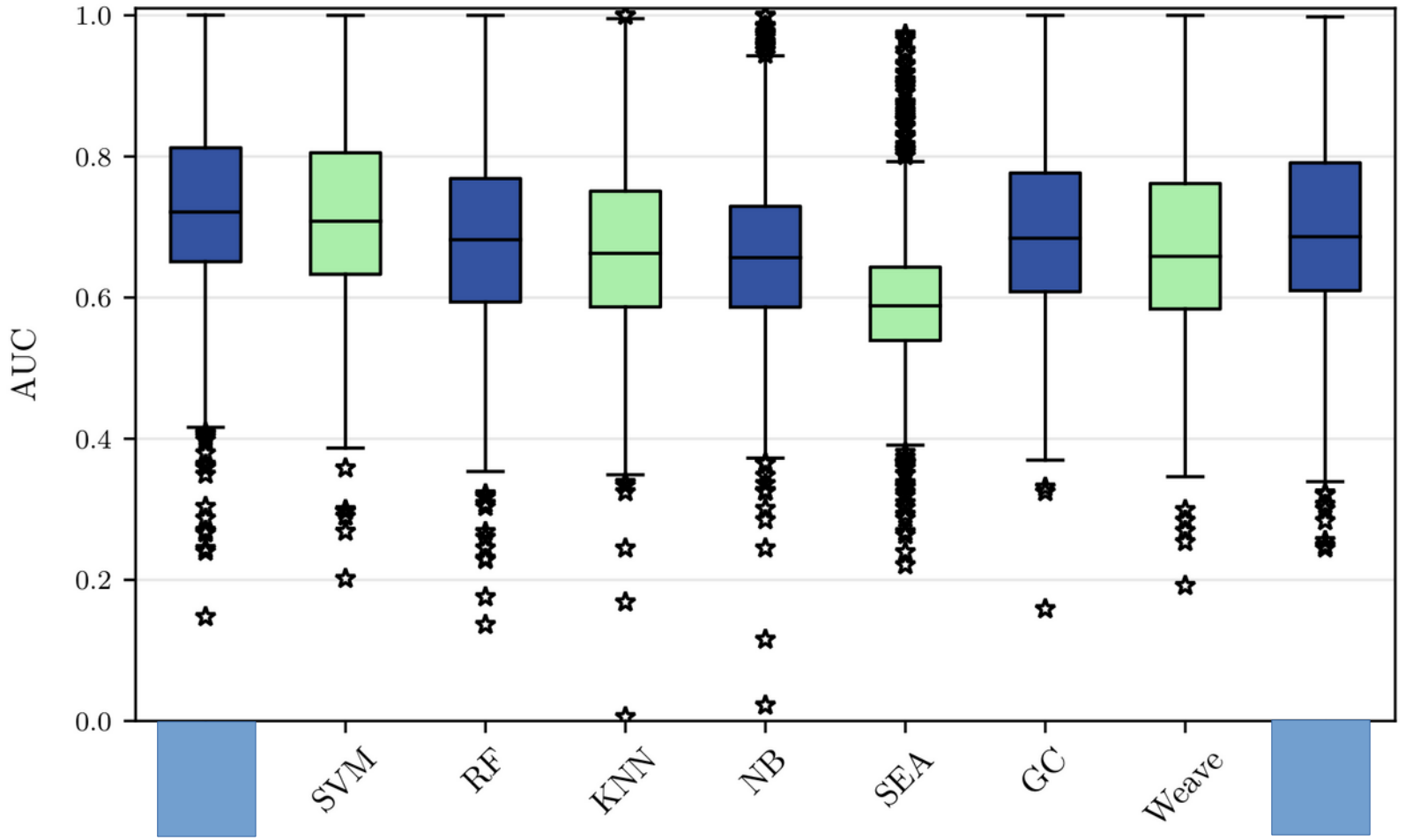
# Implementing SNNs

1) Use SELU activations
2) Initialize weights with variance 1/J, where J is the number of input neurons (" fan in”)
3) Use Alpha Dropout: dropout variant that keeps mean and variance

# Applications of self-normalizing networks (1/4): Semantic segmentation



# Applications of self-normalizing networks (2/4): Drug target prediction




Mayr, A, …, Hochreiter, S. (2018). Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. *Under review*.

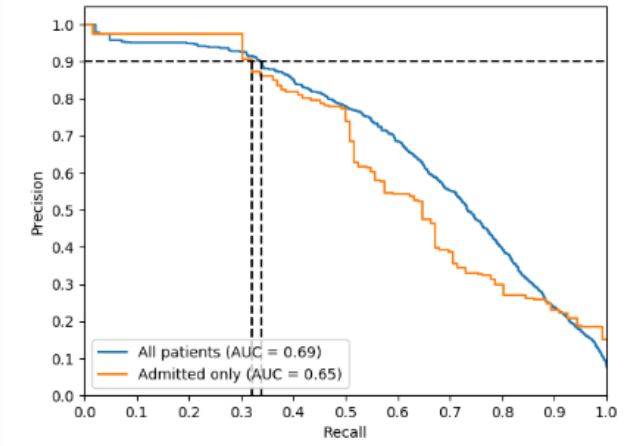
# Applications of self-normalizing networks (3/4): Deep FFN for medical and health-care applications

- Self-normalizing network with 18 layers selected
- Inputs: Patient features

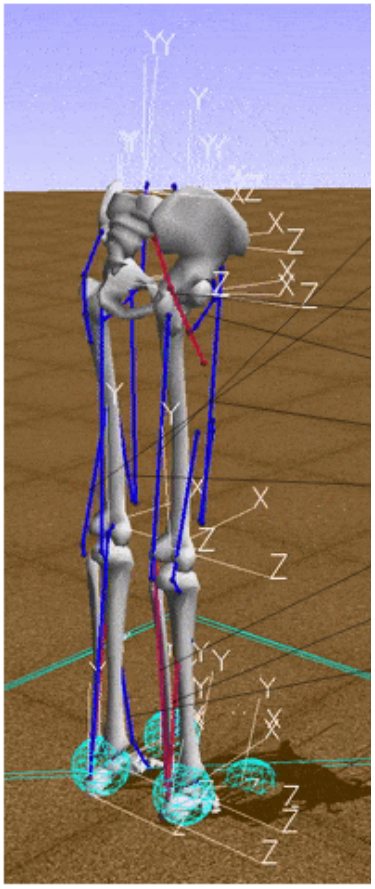
Avati, A., Jung, K., Harman, S., Downing, L., Ng, A., & Shah, N. H. (2017). Improving palliative care with deep learning. arXiv preprint arXiv:1711.06402.

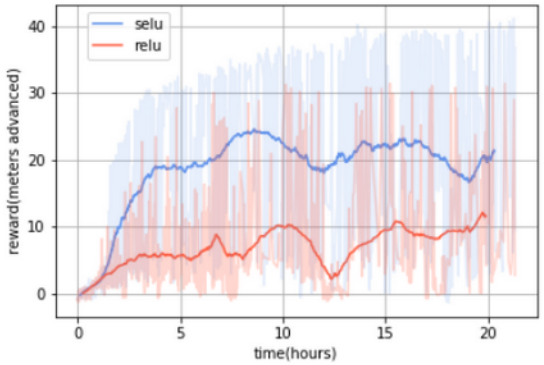
Our model is an 18-layer Deep Neural Network that inputs the EHR data of a patient, and outputs the probability of death in the next 3-12 months.

We train the model on the historic data from the Stanford Hospital EHR data base, which contains data of over 2 million patients. The model is trained to predict probability of patient mortality in the next 3-12months. Training uses patient's EHR data from the past 12 months, specifically the diagnostic codes, procedure codes, medication codes, and encounter details. All this data is converted into a feature vector for 13,654 dimensions. The trained model achieves an AUROC score of 0.93 and an Average Precision score of 0.69 on cross validation.



# Applications of self-normalizing networks (4/4): Reinforcement Learning





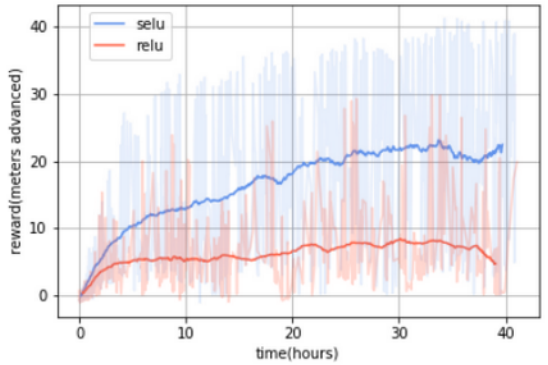




Fig. 2: Training with different activation functions and different number of processes for generating training data, by DDPG

Table 1: Hyper-parameters used in the experiments

| | |
|---|---|
| Actor network architecture | $[FC800, FC400]$, Tanh for output layer and SELU for other layers |
| Critic network architecture | $[FC800, FC400]$, linear for output layer and SELU for other layers |
| Actor learning rate | 3e-4 |
| Critic learning rate | 3e-4 |
| Batch size | 128 |
| $\gamma$ | 0.96 |
| replay buffer size | 2e6 |

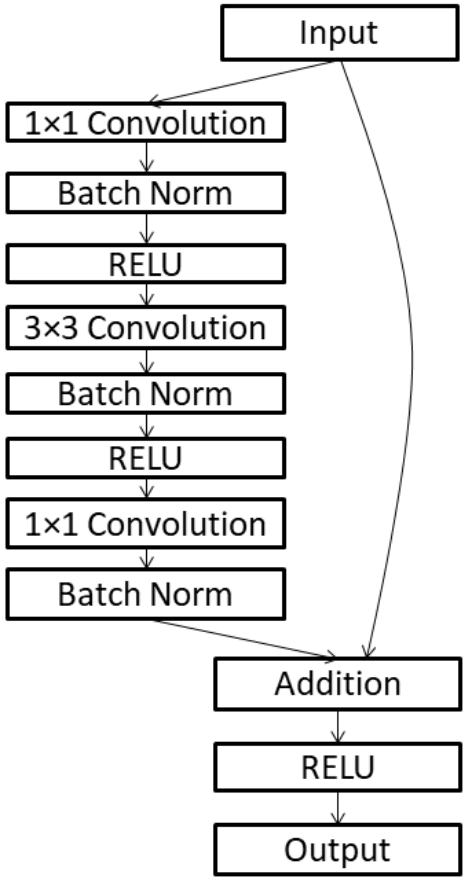
Kidziński, Ł., Mohanty, S. P., Ong, C., Huang, Z., Zhou, S., Pechenko, A., ... & Plis, S. (2018). Learning to Run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. arXiv preprint arXiv:1804.00361.

# Invariance properties of normalization techniques (Ba et al., 2016)

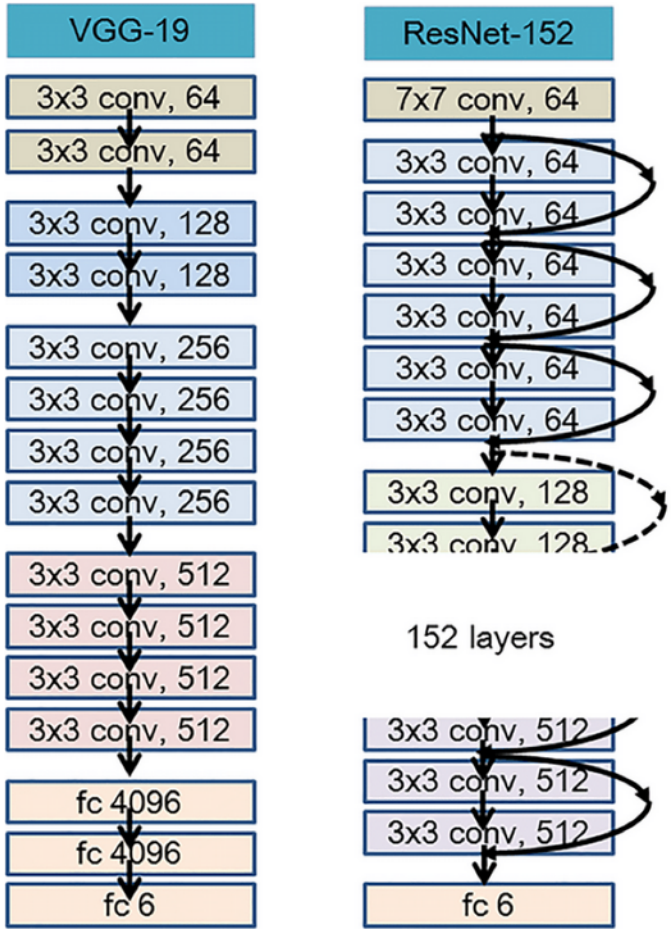
| | weight matrix re-scaling | weight matrix re-centering | weight vector re-scaling | data set re-scaling | data set re-centering | single training case re-scaling |
|---|---|---|---|---|---|---|
| batch norm | invariant | no | invariant | invariant | invariant | no |
| weight norm | invariant | no | invariant | no | no | no |
| layer norm | invariant | invariant | no | invariant | no | invariant |
| self norm | no | no | no | no | no | no |

Table 10.1: Invariance properties of normalization methods according to (Ba et al., 2016). The invariance properties only hold if the normalization technique is applied on the pre-activations of full-connected layer (linear transformation).

# Residual networks (ResNets) rely on Batch-norm




*Residual block*



# Recap

- Normalization techniques were discussed
  - Aimed at keeping activations in “good range”
    - e.g. zero mean unit variance
  - Batch-norm (most frequently used), Layer-norm, Weight-norm, Self-norm
  - Can equip networks with invariance properties
  - Main component of very deep networks (>10 layers)

# Outlook: Summer semester

- Deep Learning and Neural Nets II (365.111)
  - Advanced CNN architectures and vision applications
  - Generative and unsupervised Deep Learning
  - Theory
  - Applications
- Artificial Intelligence in Life Sciences (365.116)
  - Translation of Deep Learning techniques to problems in Life Sciences
  - Gradient: three machine learning “challenges”
- Start as “virtual lectures” due to COVID-19 measures