University of Antwerp

Artificial Neural Networks

[2500WETANN]

José Oramas

Learning & Optimization

[… for Deep Neural Networks]

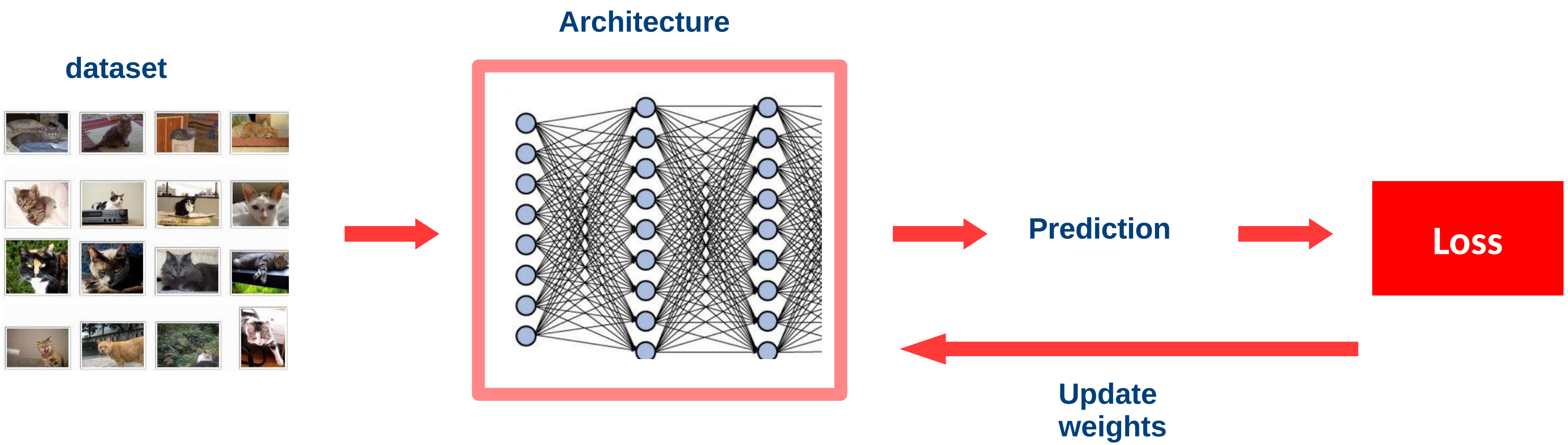
José Oramas

University of Antwerp

Learning and Optimization

Training Pipeline

- Techniques Applicable at Different Stages



Prior-Training

[The Calm Before the Storm]

Data Augmentation

What?

- Apply a set of operations on a given data sample to produce additional samples

Original Image

Cropped samples

Mirrored samples

Benefits

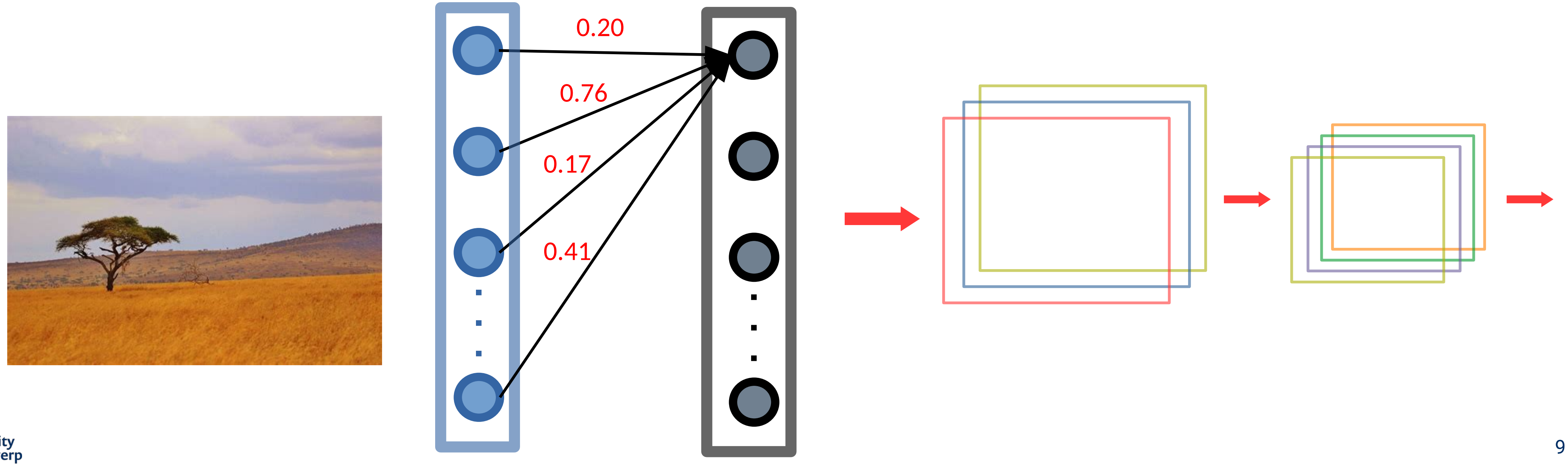
- Increase training data
- Introduce variability

Q: What happens at test time?

Input Normalization

Common Practice

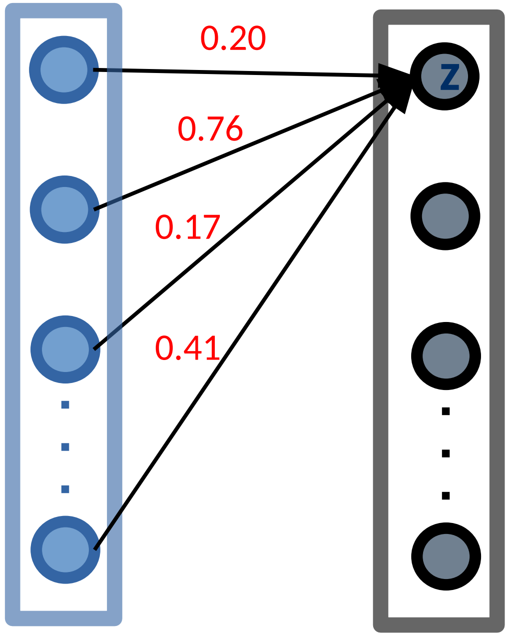
- Remove the “mean image”
- Standarize the inputs



Initializating Weights of the Network

Common Practice

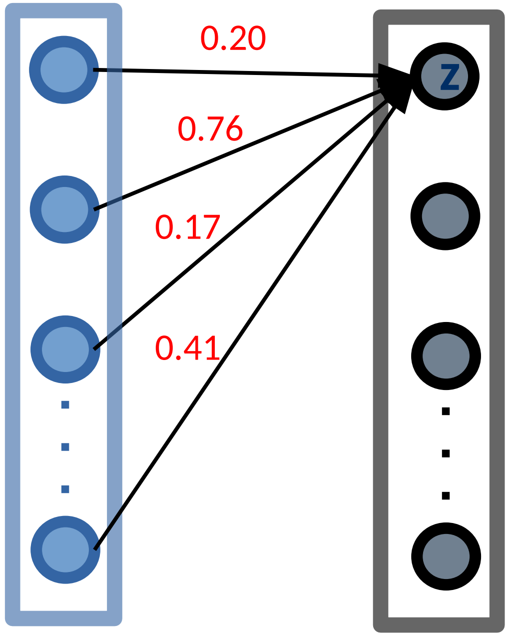
- Random Initialization
- Ensure the weights have a known mean and variance

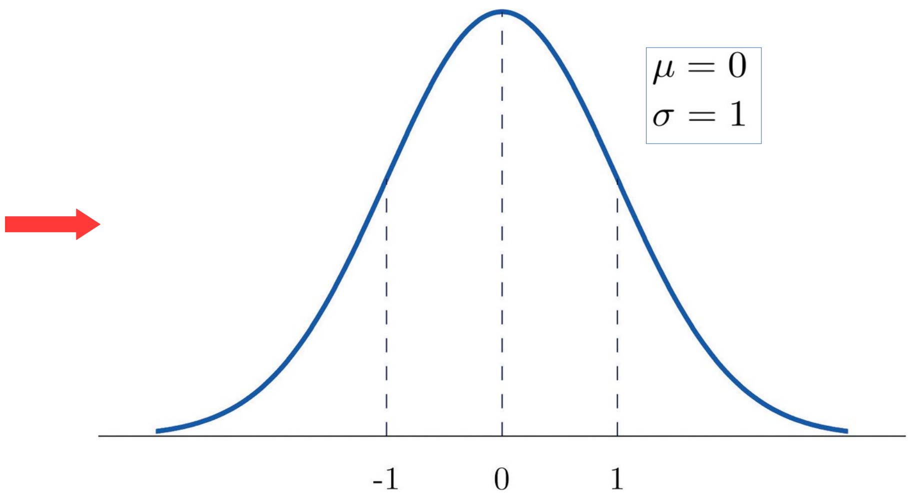


Initializating Weights of the Network

Common Practice

- Random Initialization
- Ensure the weights have a known mean and variance

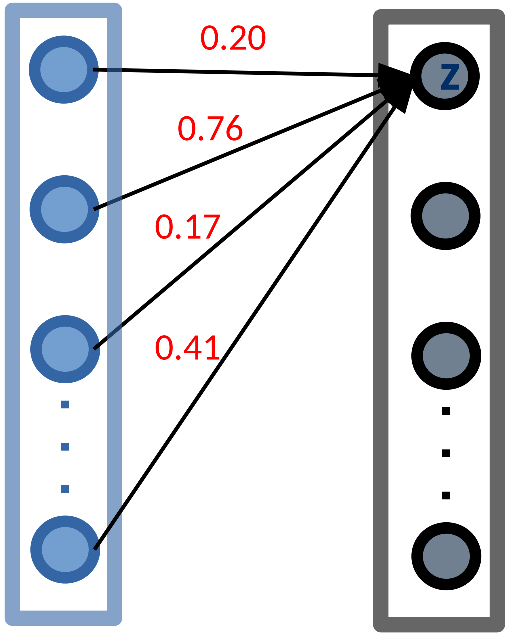


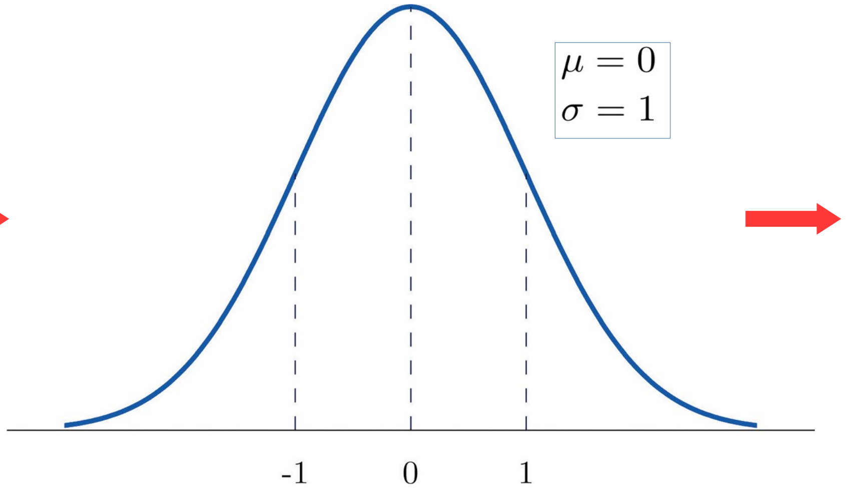


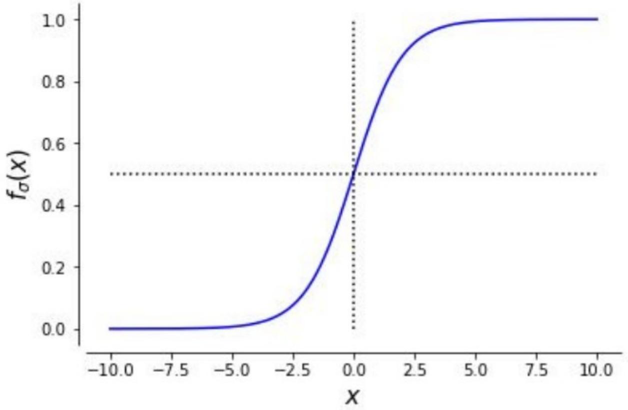
Initializating Weights of the Network

Common Practice

- Random Initialization
- Ensure the weights have a known mean and variance





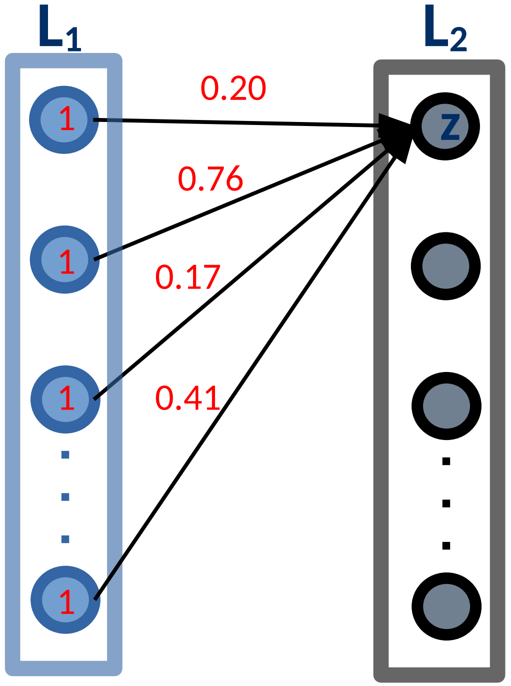


$$f_\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

Initializating Weights of the Network

Common Practice – What Happens in the Next Layer?

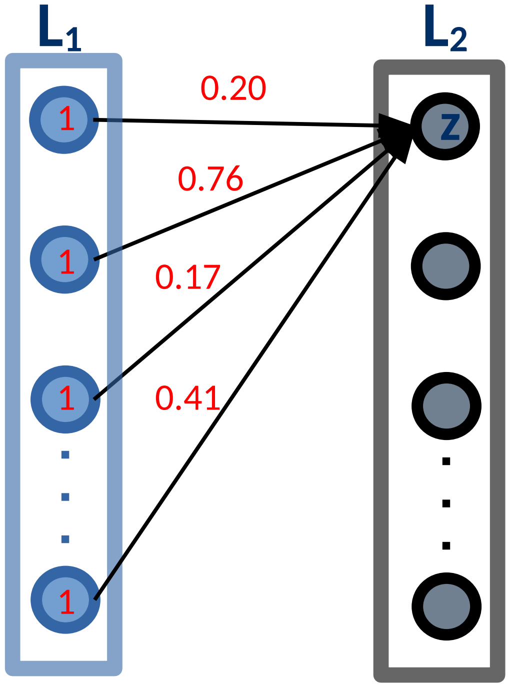
- Random Initialization
- Let's consider a simple setting (n=4 inputs, all set to 1)

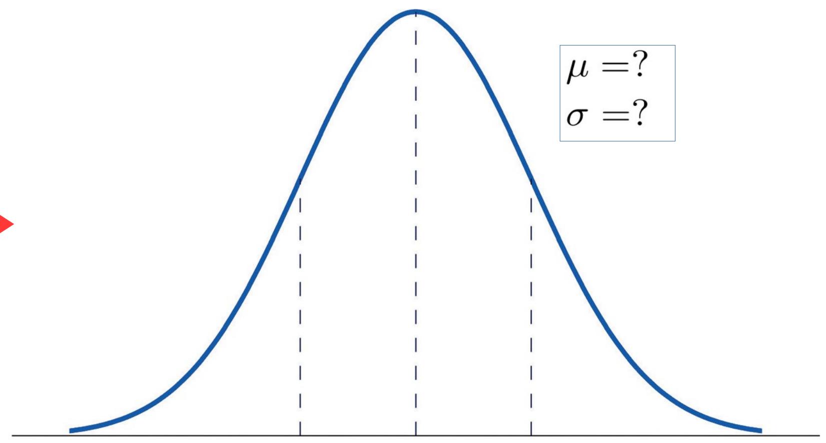


Initializating Weights of the Network

Common Practice – What Happens in the Next Layer?

- Random Initialization
- Let's consider a simple setting (n=4 inputs, all set to 1)

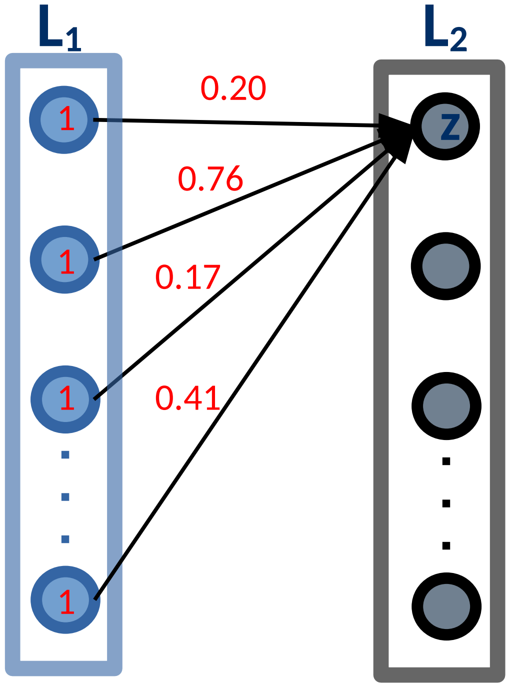


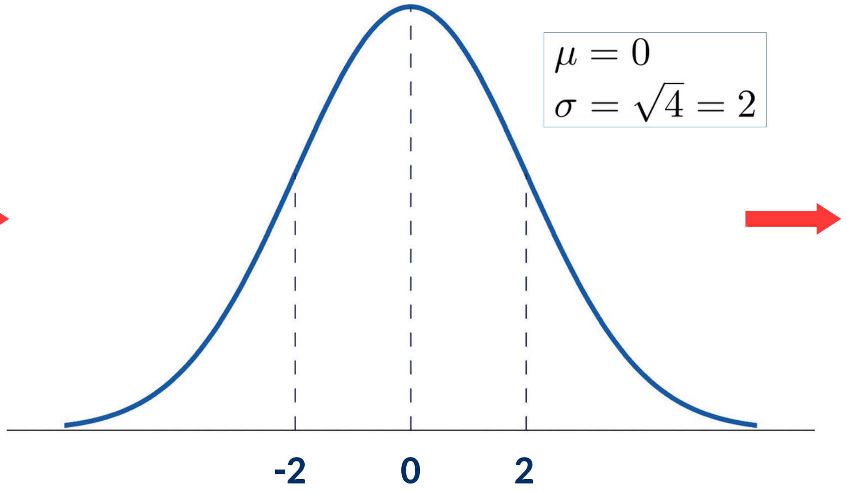


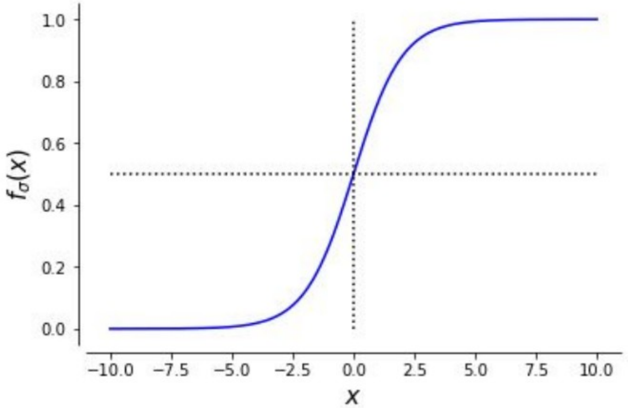
Initializating Weights of the Network

Common Practice – What Happens in the Next Layer?

- Random Initialization
- Let’s consider a simple setting (n=4 inputs, all set to 1)





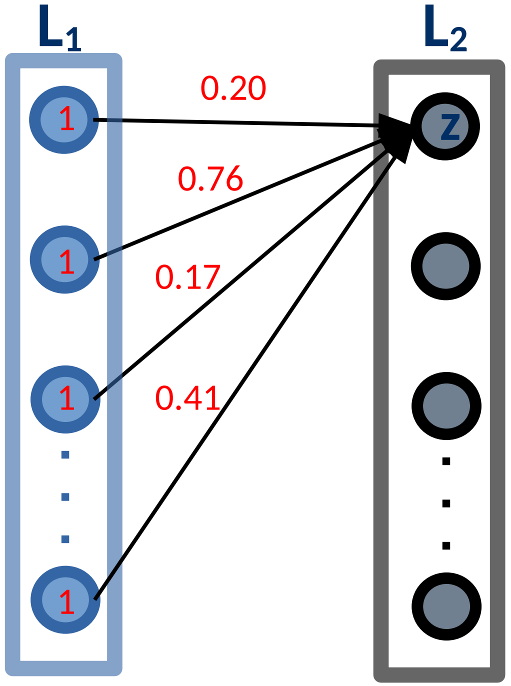


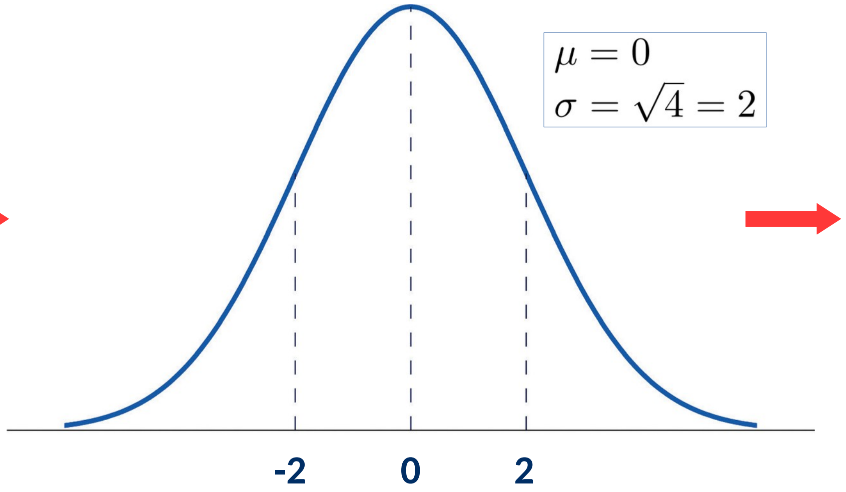
$$f_\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

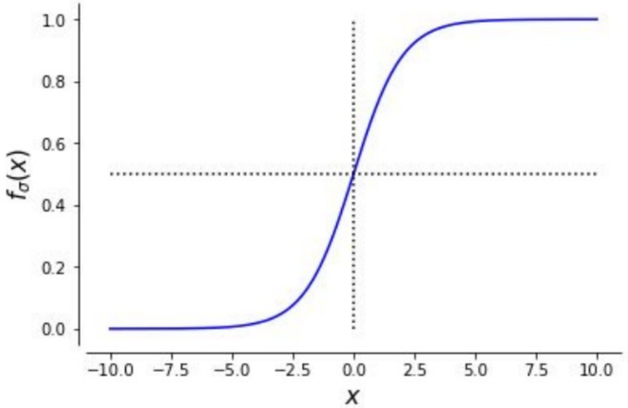
Initializating Weights of the Network

Common Practice – What Happens in the Next Layer?

- Random Initialization
- Let's consider a simple setting (n=4 inputs, all set to 1)







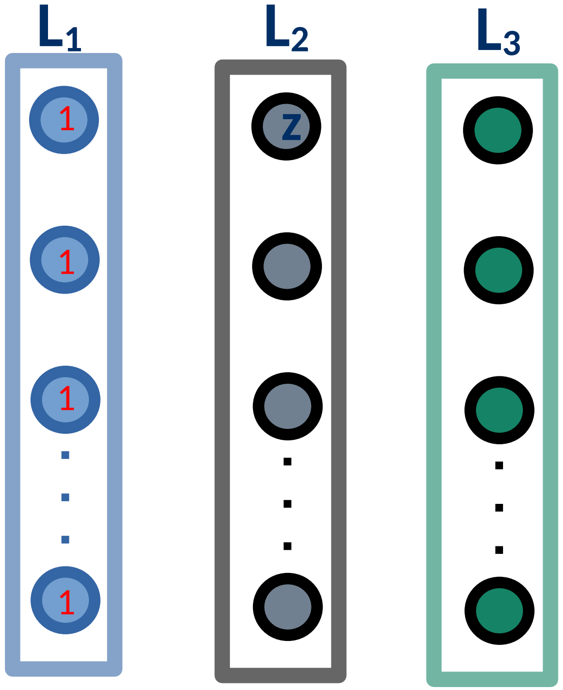
$$f_\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

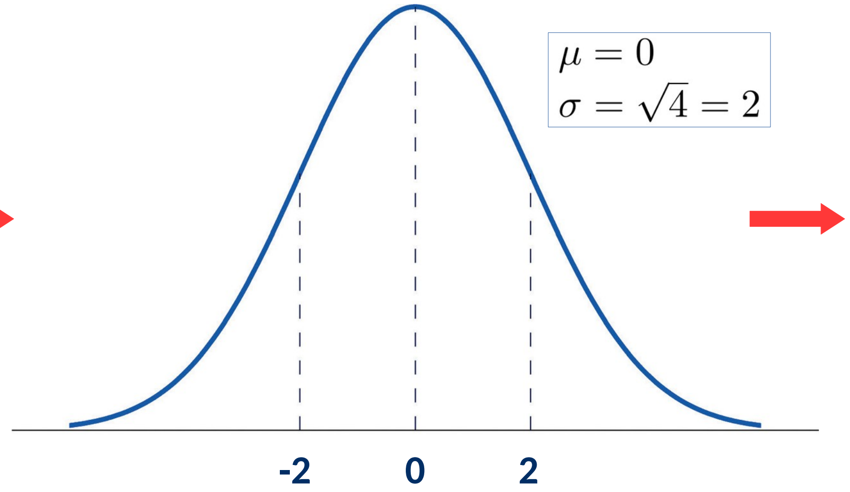
Q1: What would happen if my architecture is wider?
Q2: What would happen if my architecture is deeper?

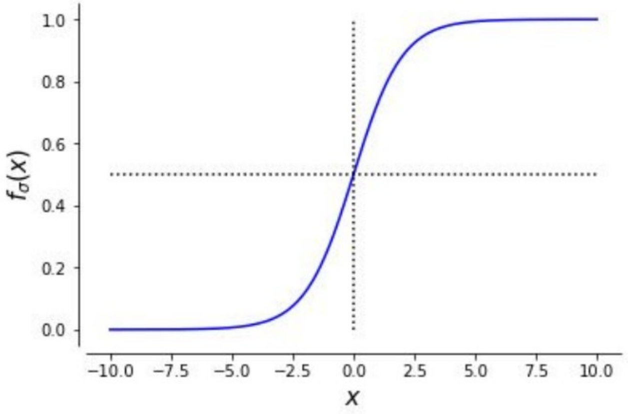
Initializating Weights of the Network

Common Practice – What Happens in the Next Layer?

- Random Initialization
- Let’s consider a simple setting (n=4 inputs, all set to 1)







$$f_\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

Q1: What would happen if my architecture is wider?
Q2: What would happen if my architecture is deeper?

Initializating Weights of the Network

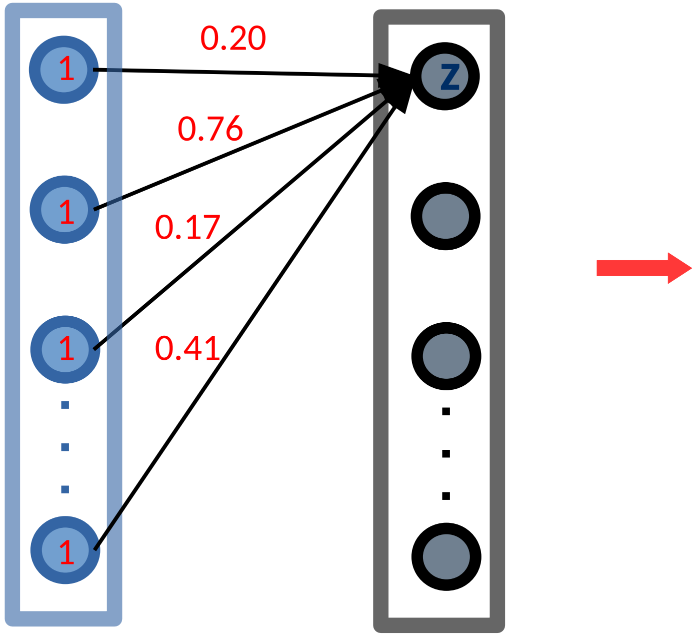
Solution – Xavier/Glorot Initialization

- **Problem:** large variance – *var(z)*
- **Solution:** let's make it smaller → *var(z) = 1/n*

Initializating Weights of the Network

Solution – Xavier/Glorot Initialization

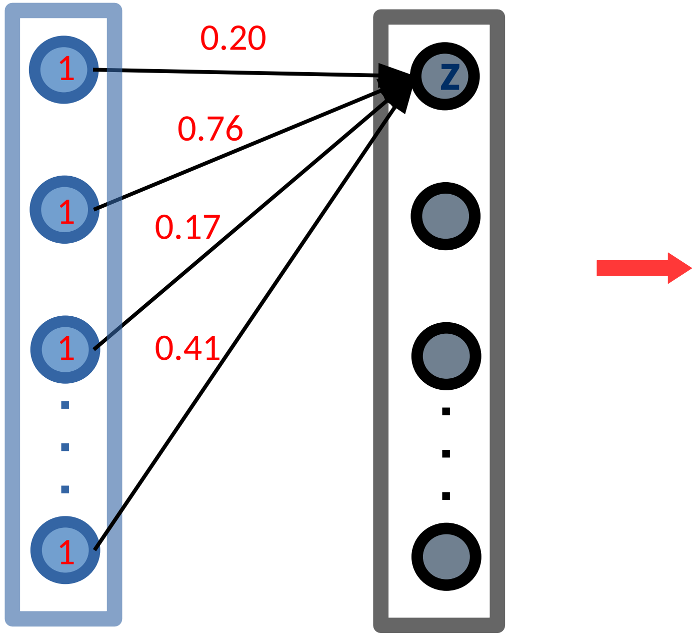
- **Problem:** large variance – *var(z)*
- **Solution:** let's make it smaller → *var(z) = 1/n*



Initializating Weights of the Network

Solution – Xavier/Glorot Initialization

- **Problem:** large variance – *var(z)*
- **Solution:** let's make it smaller → *var(z) = 1/n*



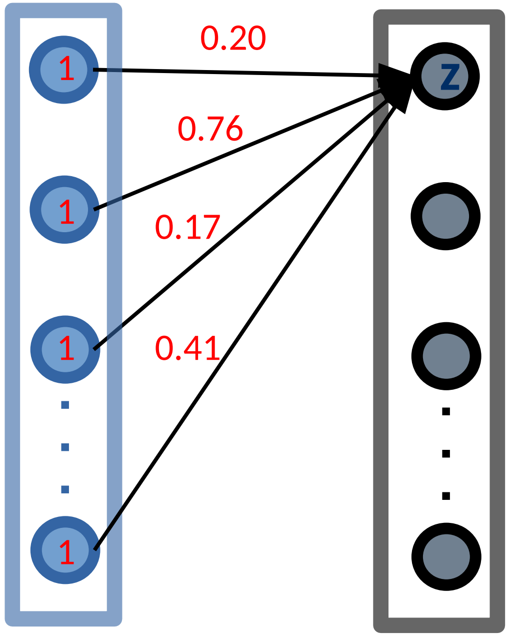
How to do it?

$$w_i = w_i \times \sqrt{1/n}$$

Initializating Weights of the Network

Solution – Xavier/Glorot Initialization

- **Problem:** large variance – *var(z)*
- **Solution:** let's make it smaller → *var(z) = 1/n*



How to do it?

$$w_i = w_i \times \sqrt{1/n}$$

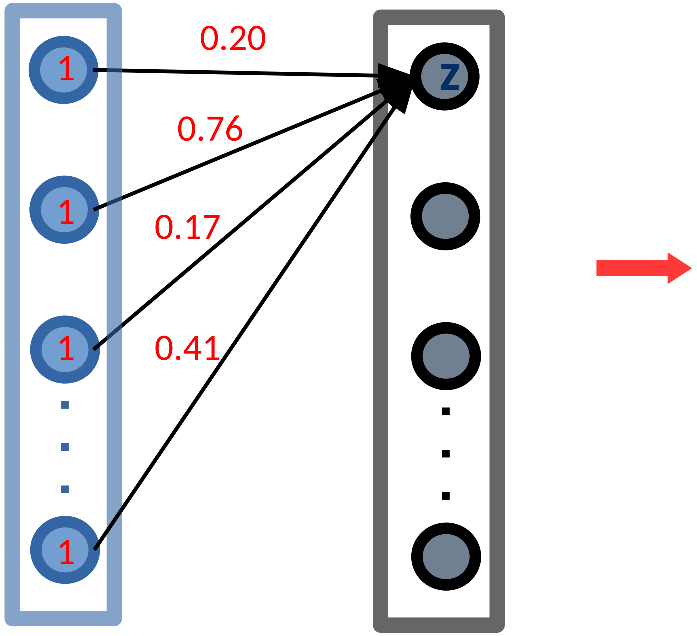
For ReLU layers

$$w_i = w_i \times \sqrt{2/n}$$

Initializating Weights of the Network

Solution – Xavier/Glorot Initialization

- **Problem:** large variance – *var(z)*
- **Solution:** let's make it smaller → *var(z) = 1/n*



How to do it?

$$w_i = w_i \times \sqrt{1/n}$$

For ReLU layers

$$w_i = w_i \times \sqrt{2/n}$$

Previously (for reference)

$$w_i = w_i \times \sqrt{\frac{2}{n_{in}+n_{out}}}$$

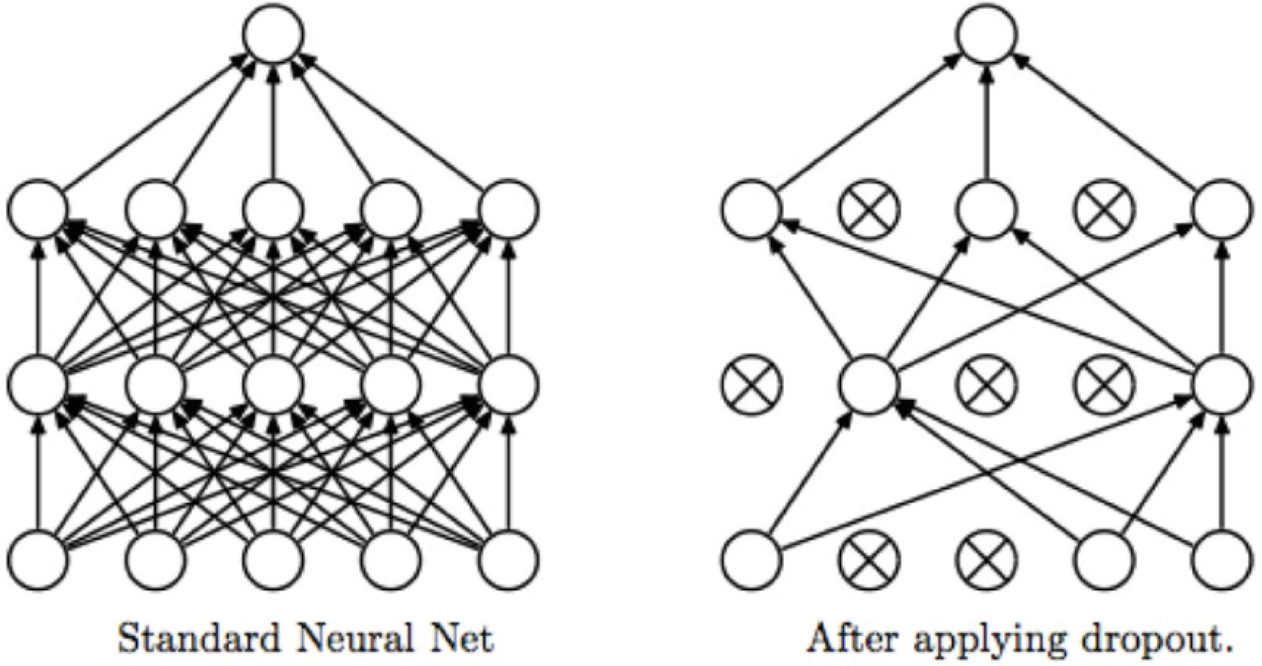
During Training

[While the Computer is Hard at Work]

Updating Weights of the Network

Dropout

- **Problem:** dicrease dependence of a given feature
- **Solution:** randomly deactivate neurons



[Krizhevsky et al., 2012]

Benefits
Avoid over-fitting
Promote ensemble learning

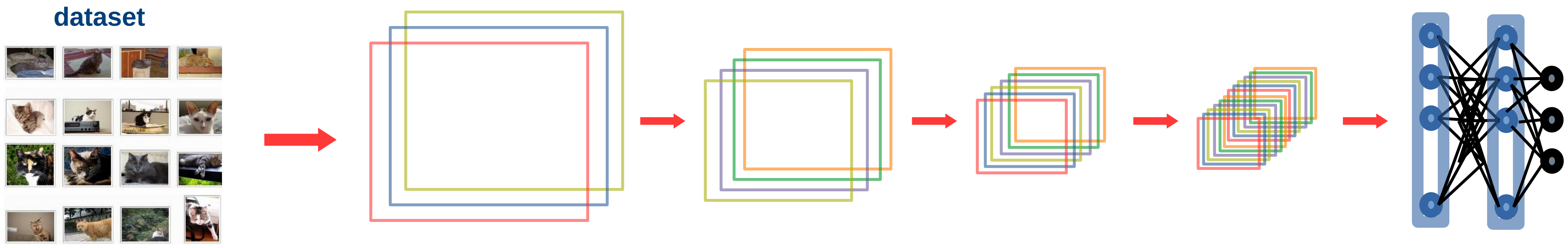
Q: What happens at test time?

University of Antwerp

Updating Weights of the Network

During Training

- **Problem:** Updates on weights at a later layers should take into account changes at earlier layers (covariance shift)
  - Introduces changes in the distribution of internal activations
  - Requires careful initialization and a small learning rate



Desiderata: Distribution of inputs (*x*) should be fixed over time

Updating Weights of the Network

Batch Normalization

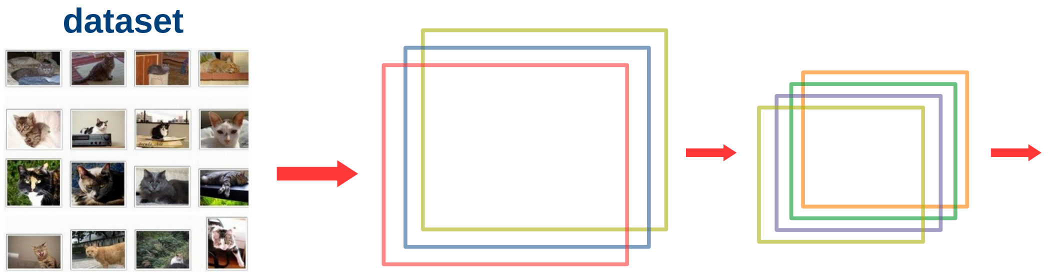
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

- **Solution:** Normalize internal acrtivations by considering dataset statistics
  - Stochastic optimization
    → batch-level statistics



[Loffe & Szegedy., 2015]

Updating Weights of the Network

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

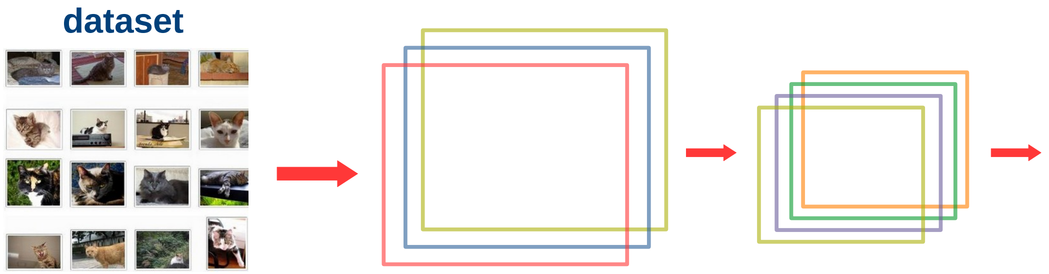
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

- **Solution:** Normalize internal acrtivations by considering dataset statistics
  - Stochastic optimization
    → batch-level statistics



[Loffe & Szegedy., 2015]

Updating Weights of the Network

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

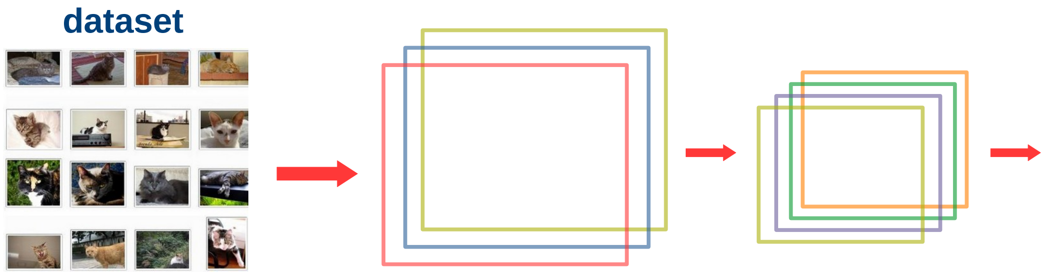
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- **Solution:** Normalize internal acrtivations by considering dataset statistics
  - Stochastic optimization

    → batch-level statistics



University of Antwerp

[Loffe & Szegedy., 2015]

Updating Weights of the Network

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

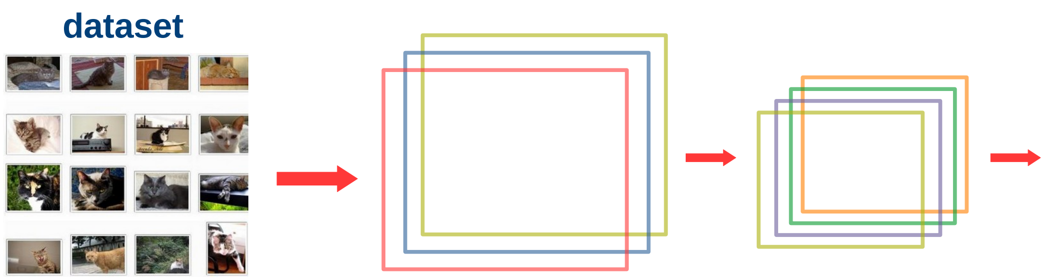
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Benefits

- Less sensitivity to initialization
- Allows using larger learning rates (faster training)



[Loffe & Szegedy., 2015]

Computing the Loss

[While Checking How Well It Works]

Break

[See you in 15 mins.]

Computing the Loss

[While Checking How Well It Works]

Training with Weighted Examples

Let's revisit the computation of the gradient of the loss

- Gradient Descend

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta L(\theta_t)$$

where,

$$\begin{aligned}\nabla_\theta L(\theta_t) &= \nabla_\theta \sum_i l(f(x^{(i)}, \theta_t), y^{(i)}) \\ &= \sum_i \nabla_\theta l(f(x^{(i)}, \theta_t), y^{(i)})\end{aligned}$$

Training with Weighted Examples

Let's revisit the computation of the gradient of the loss

- Gradient Descend

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta L(\theta_t)$$

where,

$$\nabla_\theta L(\theta_t) = \nabla_\theta \sum_i l(f(x^{(i)}, \theta_t), y^{(i)})$$

$$= \sum_i \nabla_\theta l(f(x^{(i)}, \theta_t), y^{(i)})$$

Q: Any potential problem/weakness?

University of Antwerp

Training with Weighted Examples

Let's revisit the computation of the gradient of the loss

- Gradient Descend

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta L(\theta_t)$$

where,

$$\nabla_\theta L(\theta_t) = \nabla_\theta \sum_i l(f(x^{(i)}, \theta_t), y^{(i)})$$
$$= \sum_i \nabla_\theta l(f(x^{(i)}, \theta_t), y^{(i)})$$

Q: Any potential problem/weakness?

Q: What would happen if the x_i examples are all from the same class ?

Training with Weighted Examples

Let's revisit the computation of the gradient of the loss

- Gradient Descend

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta L(\theta_t)$$

where,

$$\nabla_\theta L(\theta_t) = \nabla_\theta \sum_i l(f(x^{(i)}, \theta_t), y^{(i)})$$

$$= \sum_i \nabla_\theta l(f(x^{(i)}, \theta_t), y^{(i)})$$

re-weighting,

$$= \sum_i \frac{1}{Z(y^{(i)})} \nabla_\theta l(f(x^{(i)}, \theta_t), y^{(i)})$$

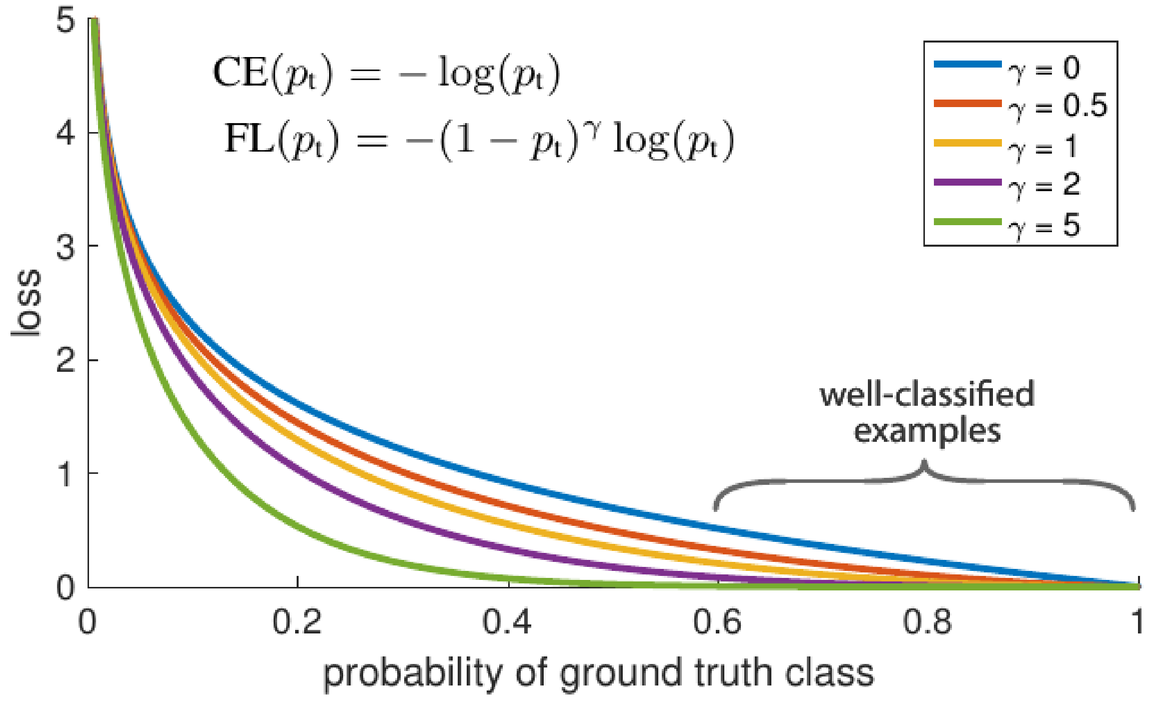
Q: Any potential problem/weakness?

Q: What would happen if the x_i examples are all from the same class ?

University of Antwerp

Training with Examples of Different Complexity

Focal Loss



$$\mathrm{FL}(p_t) = -(1-p_t)^\gamma \log(p_t)$$

What it does?

- Down-weights the loss from well-classified examples
- Focusses training on sparse set of hard examples

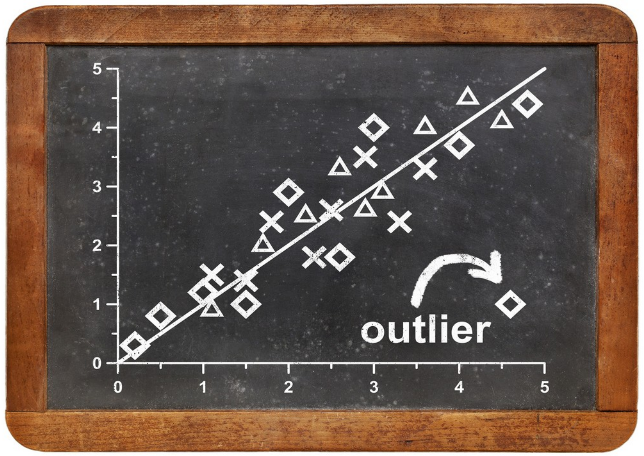
[Lin et al., 2017]

Training with Examples of Different Complexity

Focal Loss

Where it could be useful?

- Dense predictions tasks
- In the presence of outliers



Learning Representations by Comparison

Triplet Loss

- Given three examples (Anchor, Positive, Negative)
- Learn a representation that distance(positive, anchor) < distance(negative, anchor)

Learning Representations by Comparison

Triplet Loss

- Given three examples (Anchor, Positive, Negative)
- Learn a representation that distance(positive, anchor) < distance(negative, anchor)

Definition

$$L(A,P,N) = max(\ d(A,P) - d(A,N) + \alpha\ ,\ 0\ )$$

Learning Representations by Comparison

Triplet Loss

- Given three examples (Anchor, Positive, Negative)
- Learn a representation that distance(positive, anchor) < distance(negative, anchor)

Definition

$$L(A, P, N) = max(\ d(A, P) - d(A, N) + \alpha\ ,\ 0\ )$$

Using the L2-distance

$$L(A, P, N) = max(\ ||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha\ ,\ 0\ )$$

Learning Representations by Comparison

Triplet Loss

- Given three examples (Anchor, Positive, Negative)
- Learn a representation that distance(positive, anchor) < distance(negative, anchor)

Definition

$$L(A,P,N) = max(\ d(A,P) - d(A,N) + \alpha\ ,\ 0\ )$$

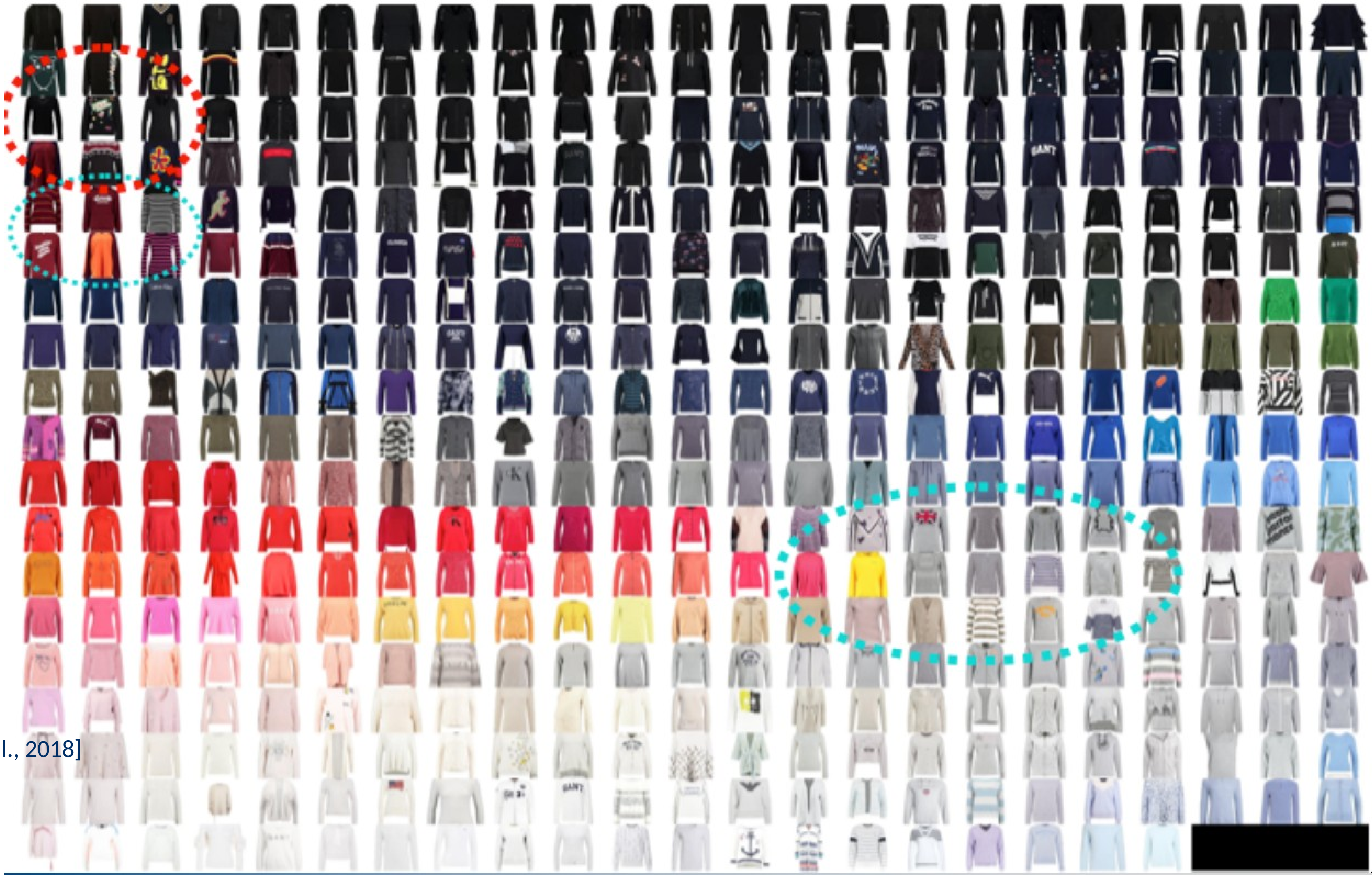
Using the L2-distance

$$L(A,P,N) = max(\ ||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha\ ,\ 0\ )$$

Using in a cost function $= \sum_{i=1}^{M} L(\ A^{(i)}, P^{(i)}, N^{(i)}\ )$

Triplet Loss

[Wang et al., 2018]

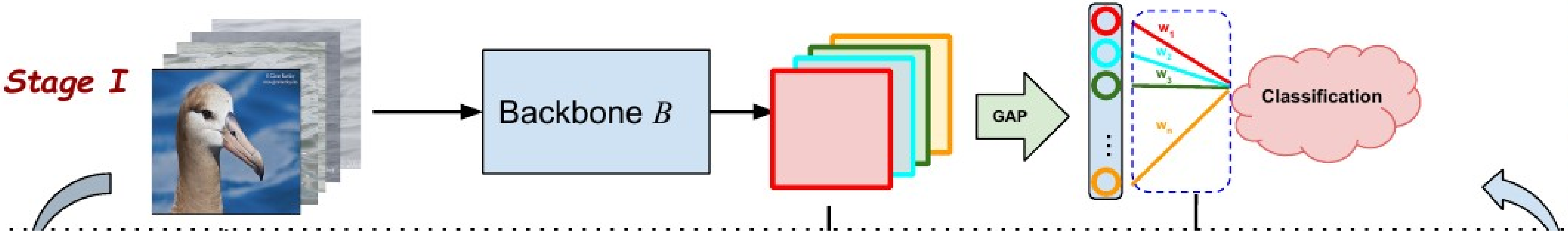


[Wang et al., 2018]

Using Multiple Loss Functions

Object Localization – MinMaxCAM [Wang et al., 2021]

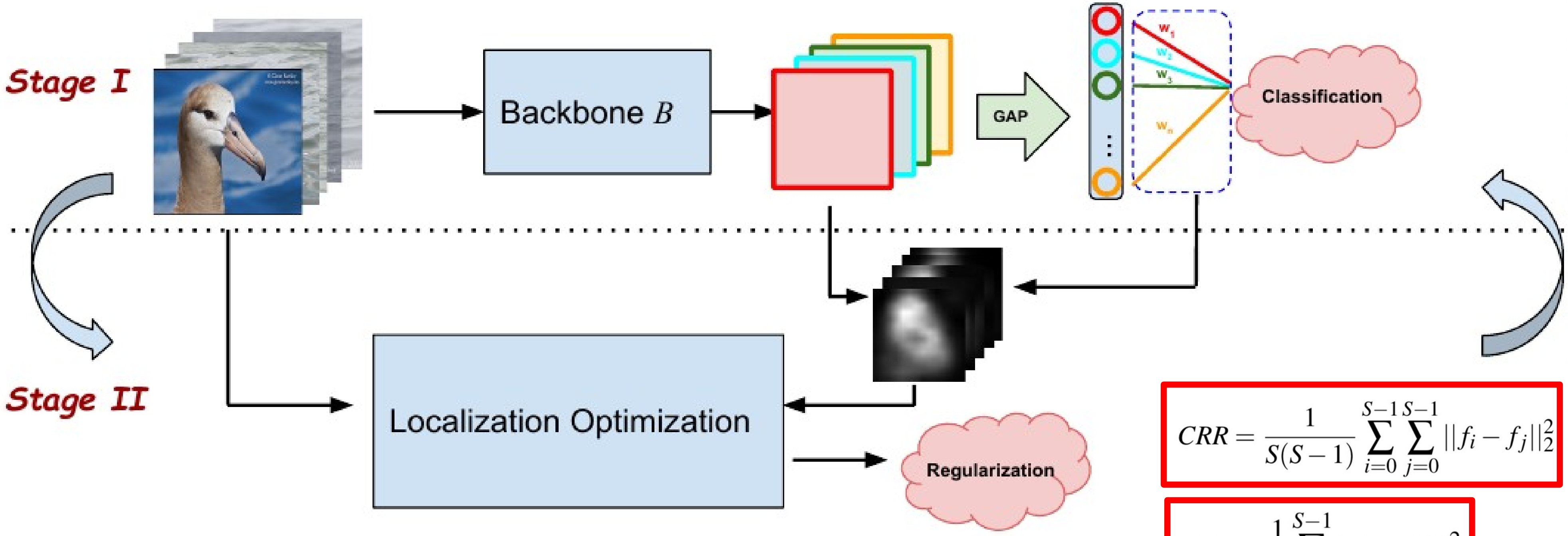
Idea: Regularize a high-performing classifier to enable localization



Using Multiple Loss Functions

Object Localization – MinMaxCAM [Wang et al., 2021]

Idea: Regularize a high-performing classifier to enable localization



$$CRR = \frac{1}{S(S-1)} \sum_{i=0}^{S-1} \sum_{j=0}^{S-1} ||f_i - f_j||_2^2$$

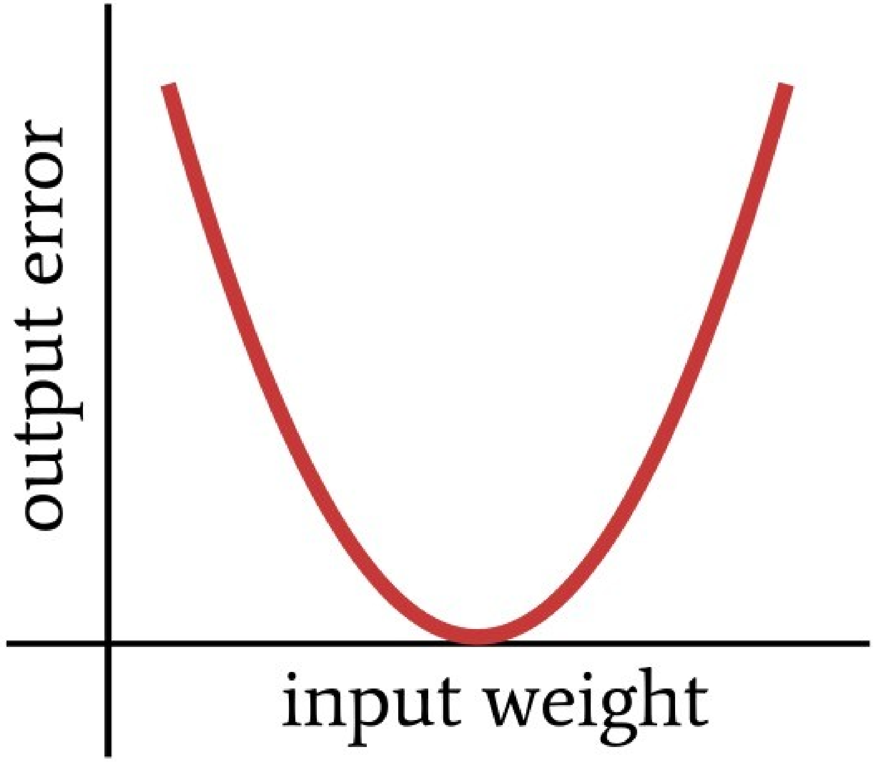
$$FRR = \frac{1}{S} \sum_{i=0}^{S-1} ||f_i - f_i^o||_2^2$$

Optimization

[While Searching for the Best Solution]

Optimizing the Training Procedure

Fixed VS Variable Learning Rate

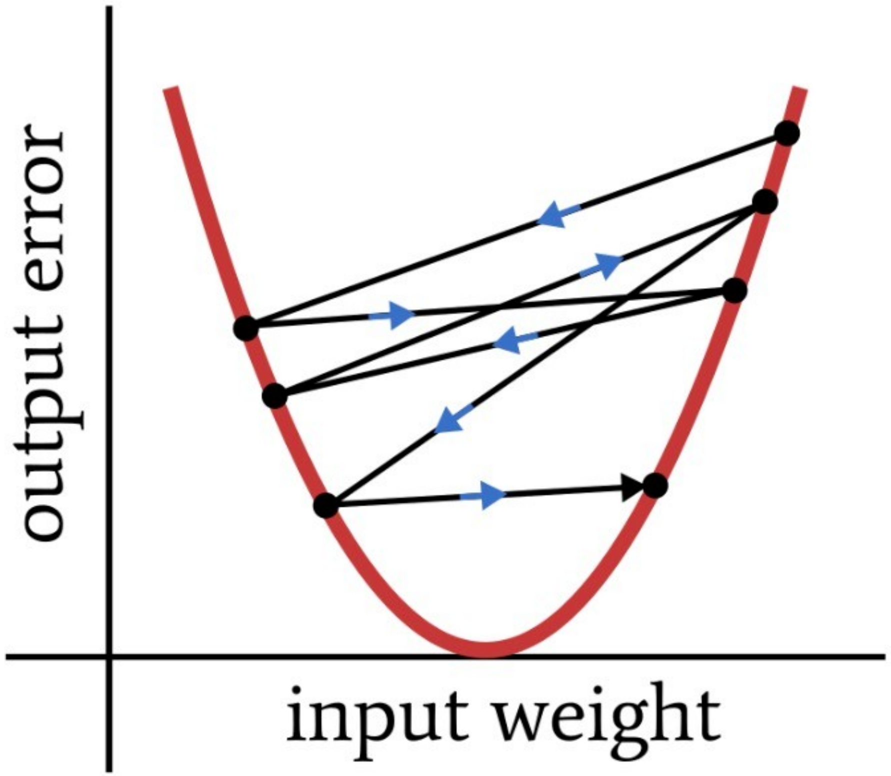


$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta L(\theta_t)$$

- **Problem:** As training progresses taken steps might be to large to reach the optimum

Optimizing the Training Procedure

Fixed VS Variable Learning Rate

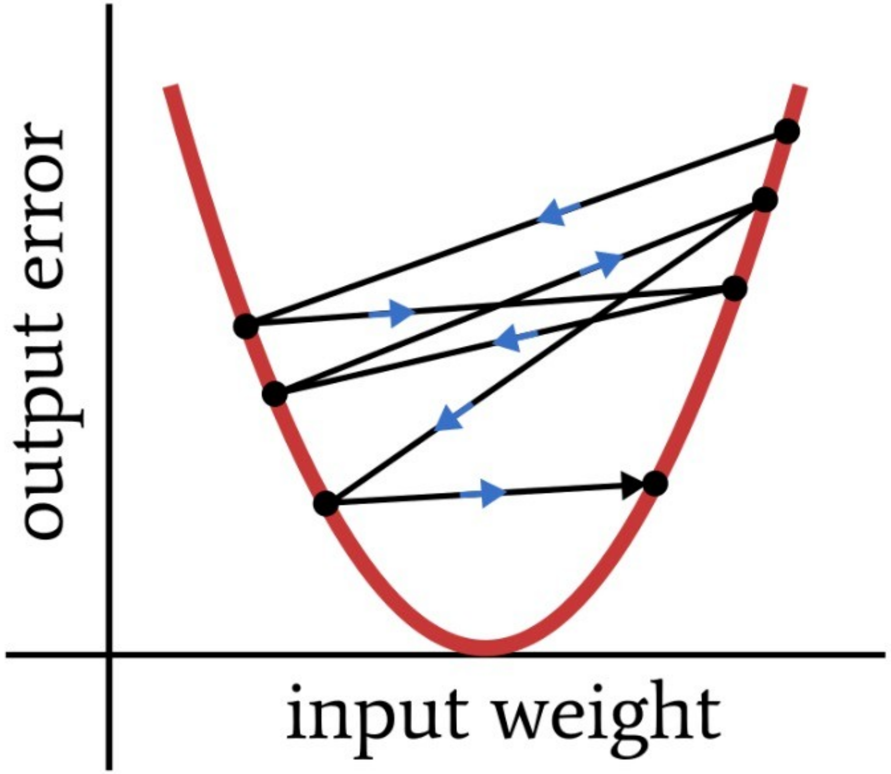


$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta L(\theta_t)$$

- **Problem:** As training progresses taken steps might be to large to reach the optimum

Optimizing the Training Procedure

Fixed VS Variable Learning Rate



$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta L(\theta_t)$$

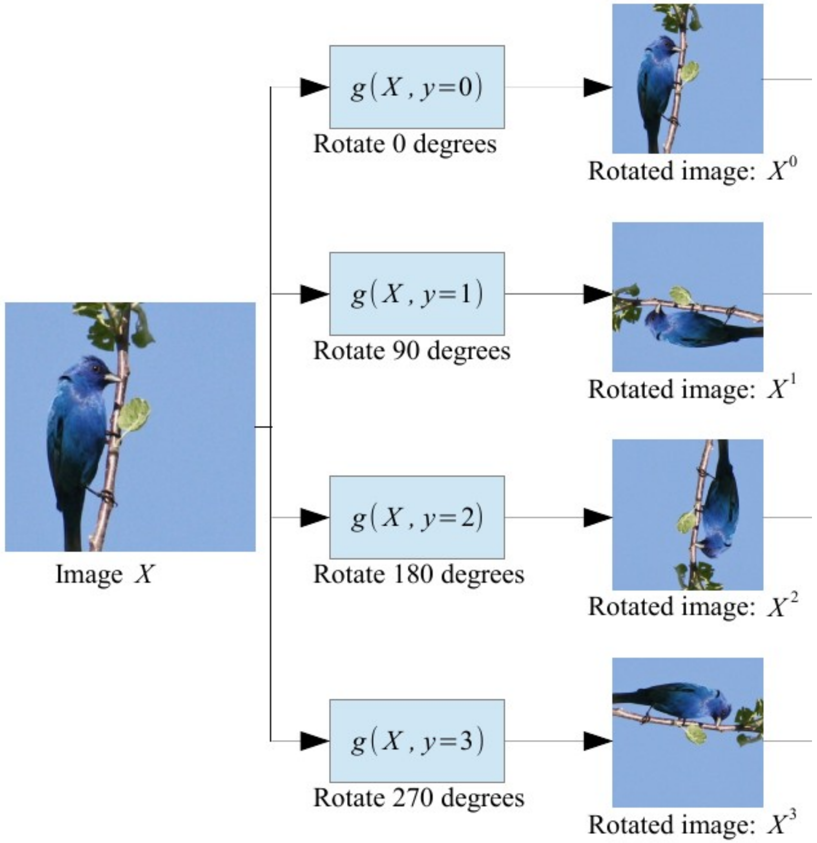
- **Problem:** As training progresses taken steps might be to large to reach the optimum
- **Solution:** dicrease the learning rate as training progresses.
  (Annealing)

Combinations

[A bit of everything]

Self-Supervised Learning

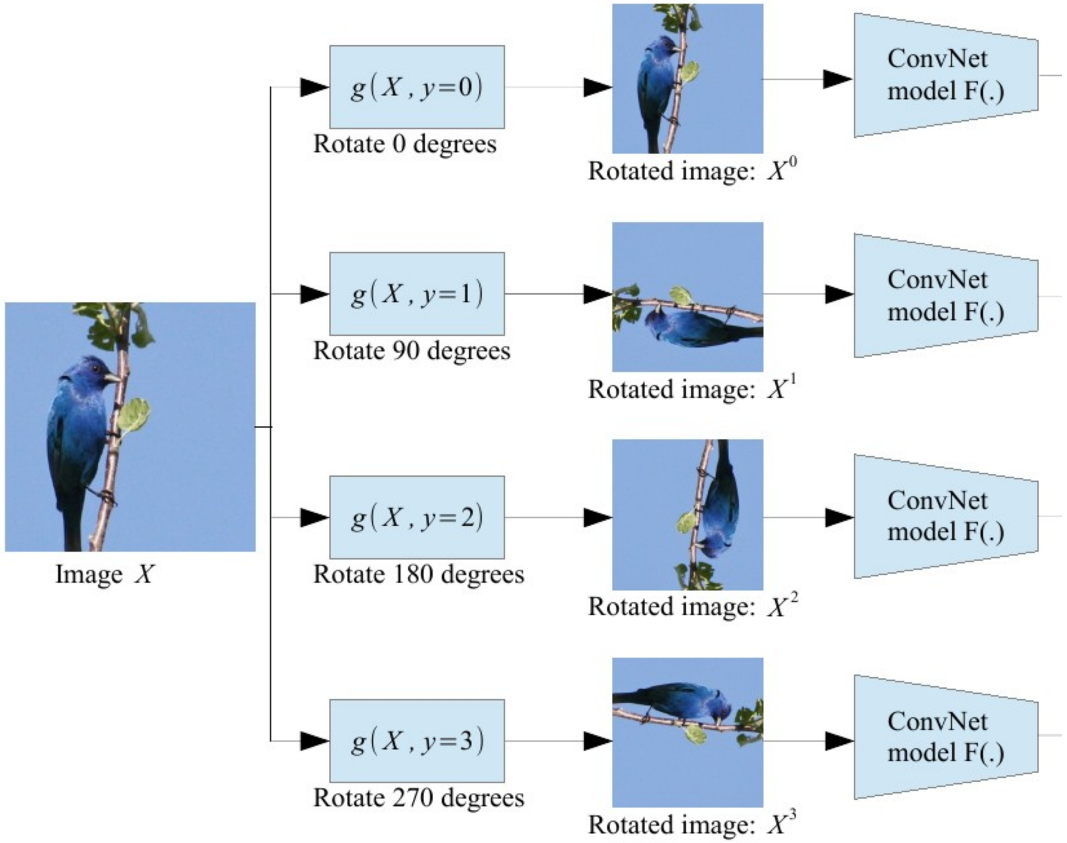
- Problem: data annotation is expensive
- Solution: supervise using labels generated from data (without manual annotation)



[Gidaris et al., 2018]

Self-Supervised Learning

- Problem: data annotation is expensive
- Solution: supervise using labels generated from data (without manual annotation)

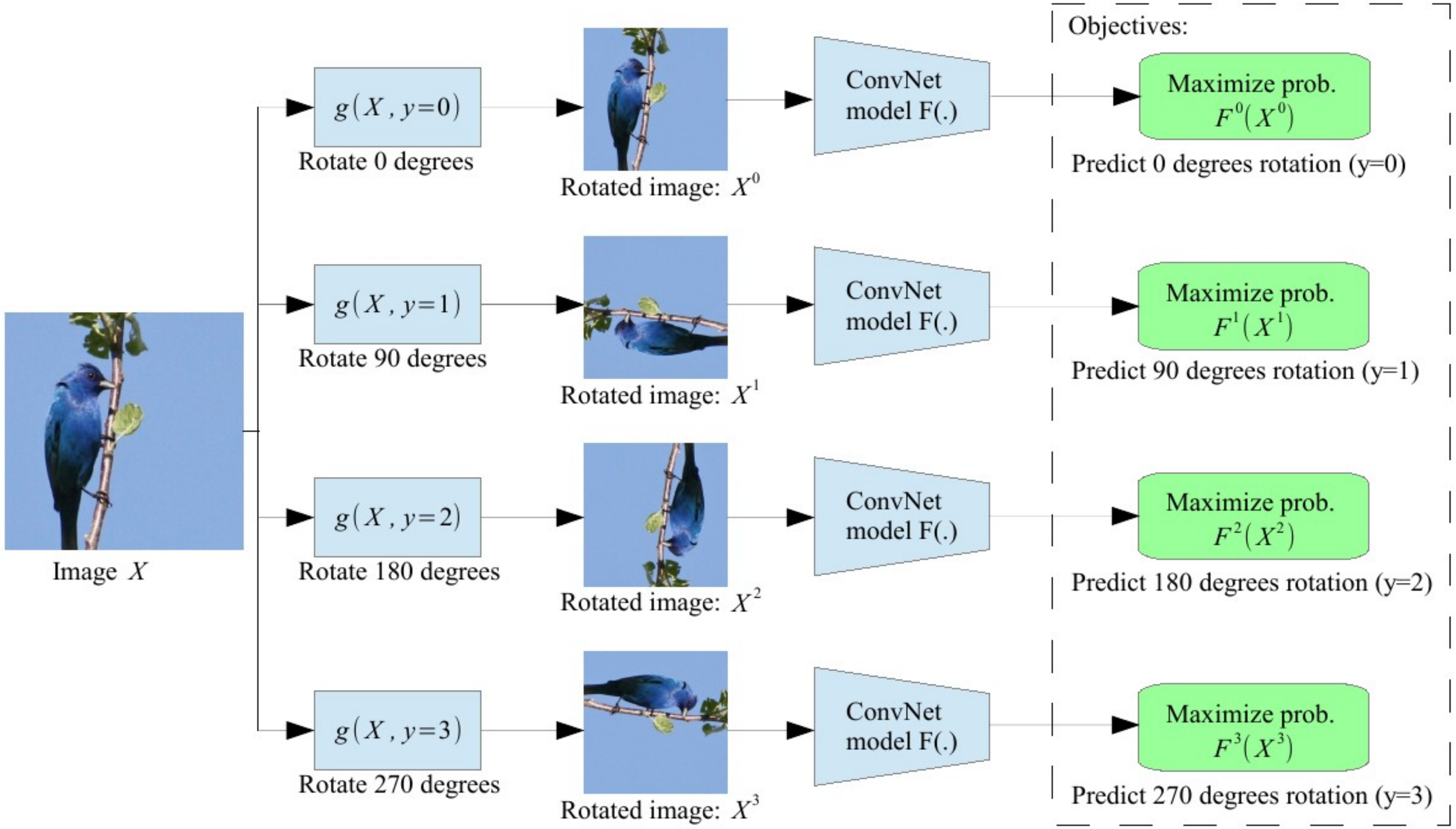


[Gidaris et al., 2018]

Self-Supervised Learning

- Problem: data annotation is expensive
- Solution: supervise using labels generated from data (without manual annotation)

Pretext task

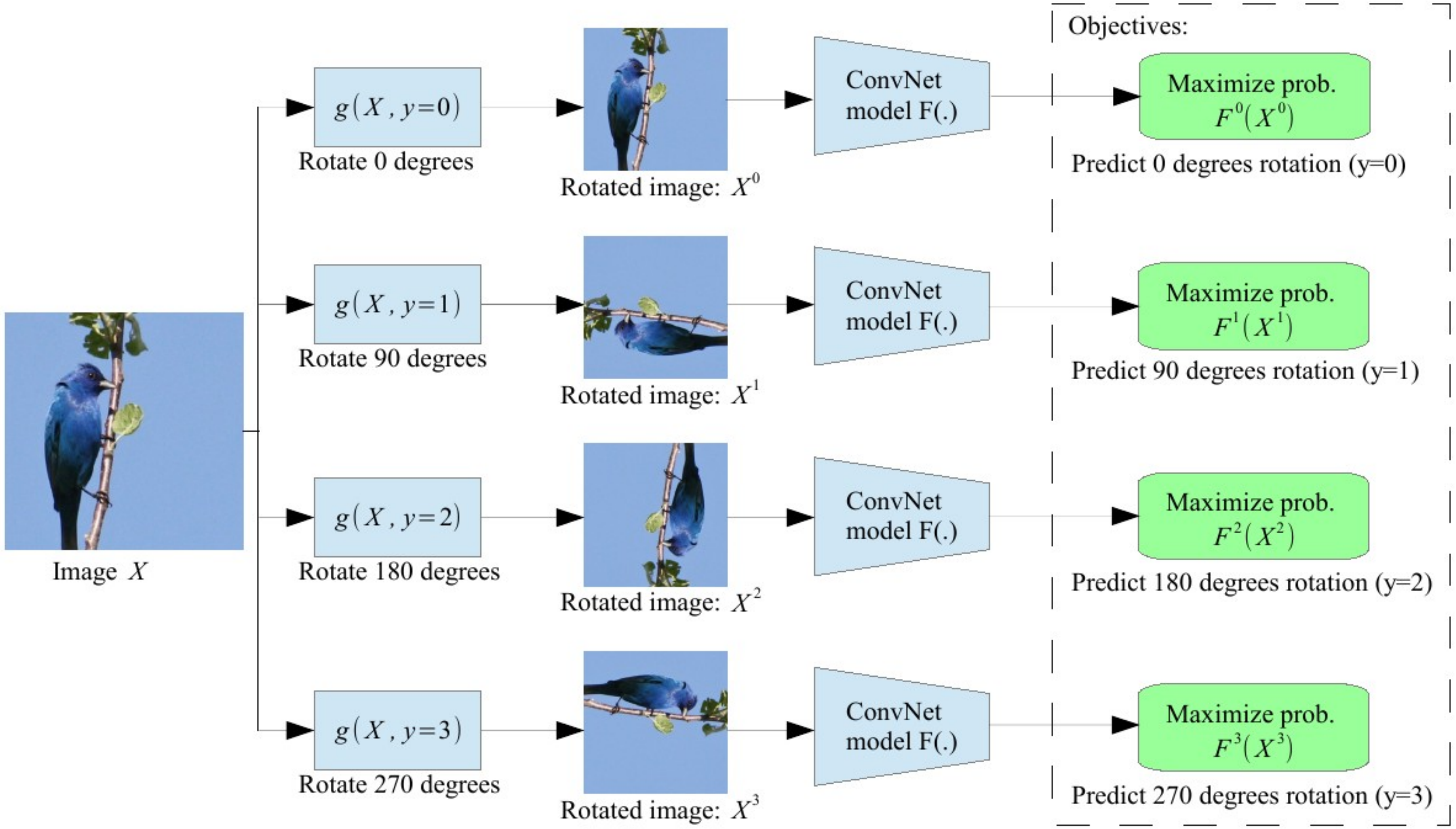


[Gidaris et al., 2018]

Self-Supervised Learning

- Problem: data annotation is expensive
- Solution: supervise using labels generated from data (without manual annotation)

Pretext task



[Gidaris et al., 2018]

University of Antwerp

Summarizing

[Finally :D]

Summarizing

- Different techniques possible

    Additional adaptation to the problem at hand might be required

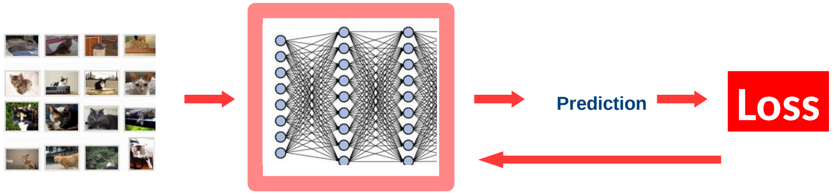
Summarizing

- **Different techniques possible**

  Additional adaptation to the problem at hand might be required

- **Can be applied at different stages of training**

  pre/during training | computing the loss | updating the weights



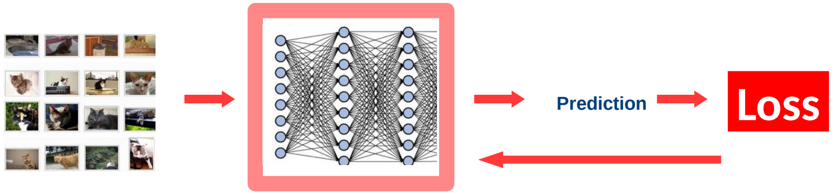
Summarizing

- **Different techniques possible**

  Additional adaptation to the problem at hand might be required

- **Can be applied at different stages of training**

  pre/during training | computing the loss | updating the weights



- **Nothing but the tip of very huge iceberg**

  lots of other techniques available

Pay Attention to...

- When the techniques are applied?
  - Additional actions to be taken at different times
- What is the problem they address?
- How these techniques operate and what is their effect
  - How to adapt them to specific architectures?

Questions?

References

- S. Ioffe and C. Szegedy

  Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. ICML 2015.

- A. Krizhevsky, I. Sutskever, G. E. Hinton.

  ImageNet Classification with Deep Convolutional Neural Networks. NeurIPS 2012

- T. Lin, P. Goyal, R. Girshick, K. He, P. Dollár

  Focal Loss for Dense Object Detection. ICCV 2017

- K. Wang, L. Ma, J. Oramas, L. Van Gool and T. Tuytelaars

  Unsupervised shape transformer for image translation and cross-domain retrieval. ArXiv:1812.02134, 2018

- S. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh,

  Convolutional Pose Machines, CVPR 2016.

- Spyros Gidaris, Praveer Singh, Nikos Komodakis.

  Unsupervised Representation Learning by Predicting Image Rotations. ICLR 2018

University of Antwerp

Artificial Neural Networks

[2500WETANN]

José Oramas