Machine Learning course
advanced track

CNN Training Tricks

Iurii Efimov

MIPT
2021, Moscow, Russia

Outline

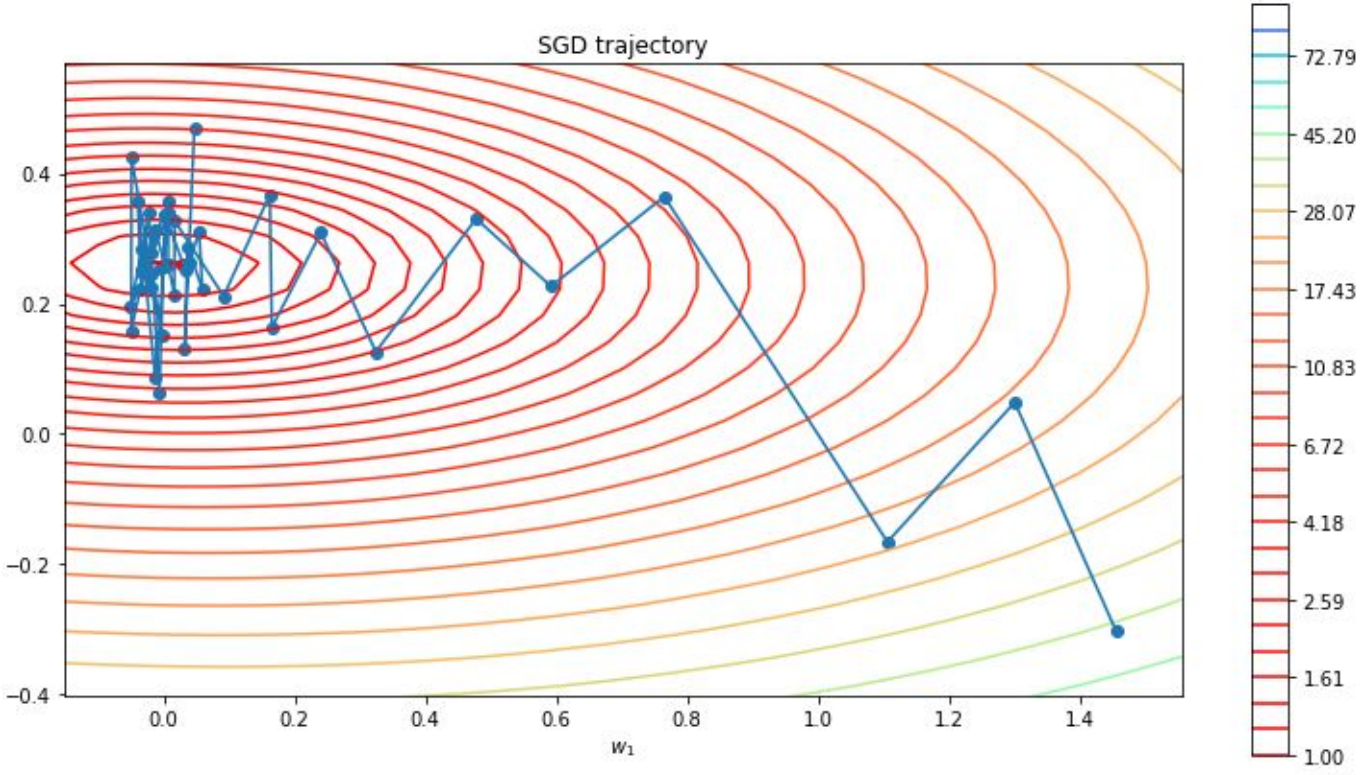
- Learning rate schedulers
- Learning rate warmup
- Regularization: Label smoothing, Mixup, DropBlock
- Contrastive training

Learning Rate

Learning rate schedulers

Motivation:

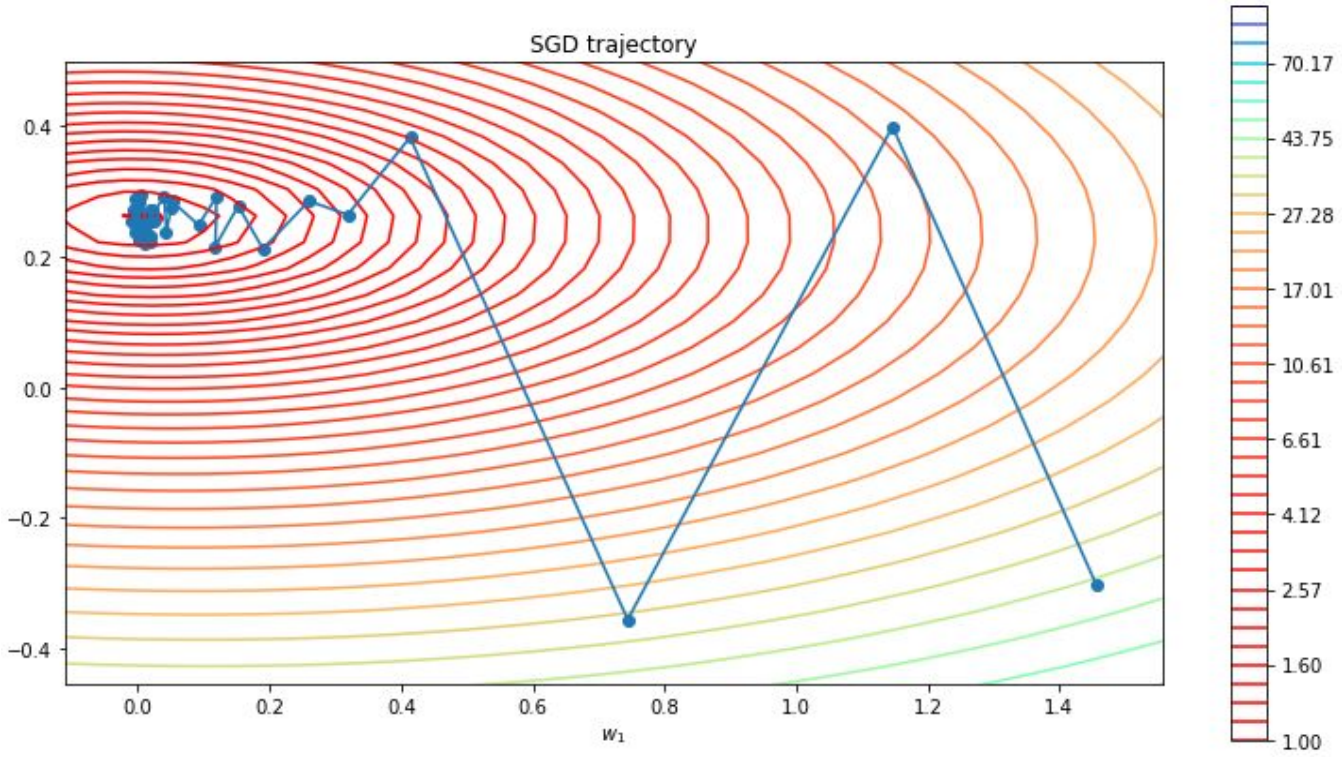
- constant learning rate → constant weight updates
- early training stages: OK
- later training stages: may miss global minima due to large weight updates



Learning rate schedulers

Motivation:

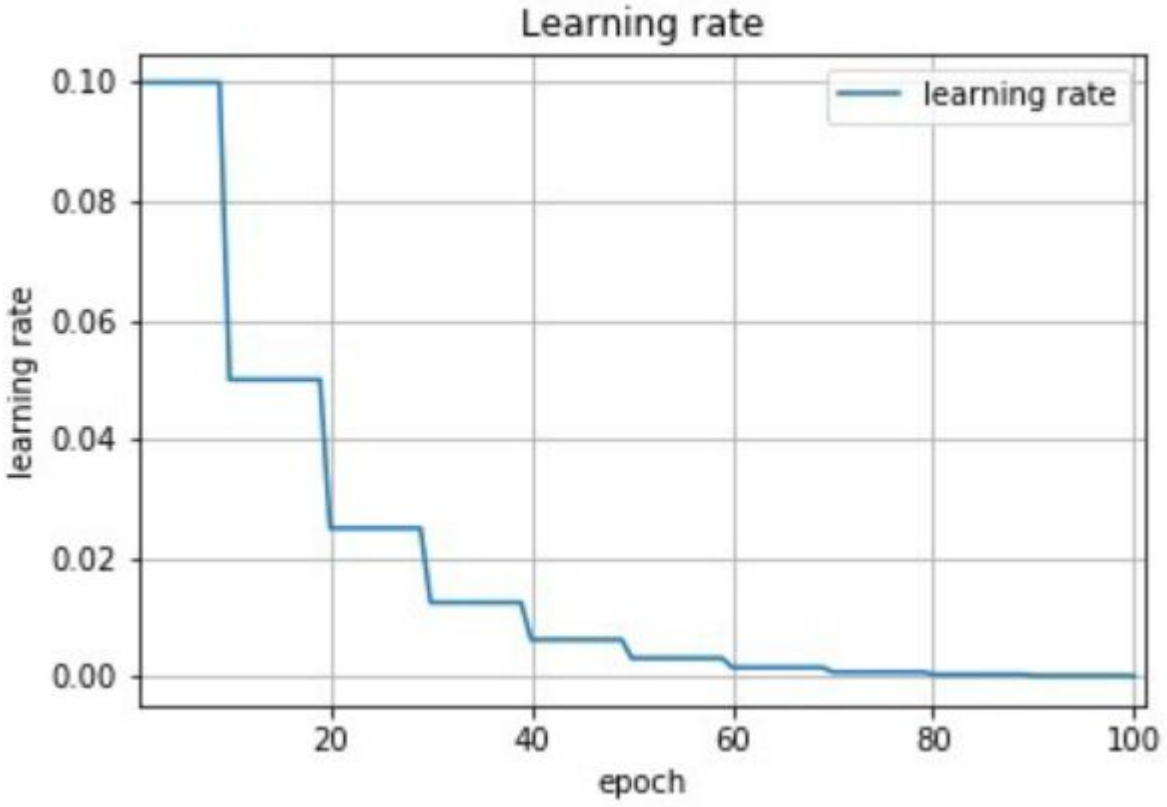
- adaptive learning rate → smaller weight updates
- usual strategy: decreasing learning rate during training



Learning rate schedulers

Step decay

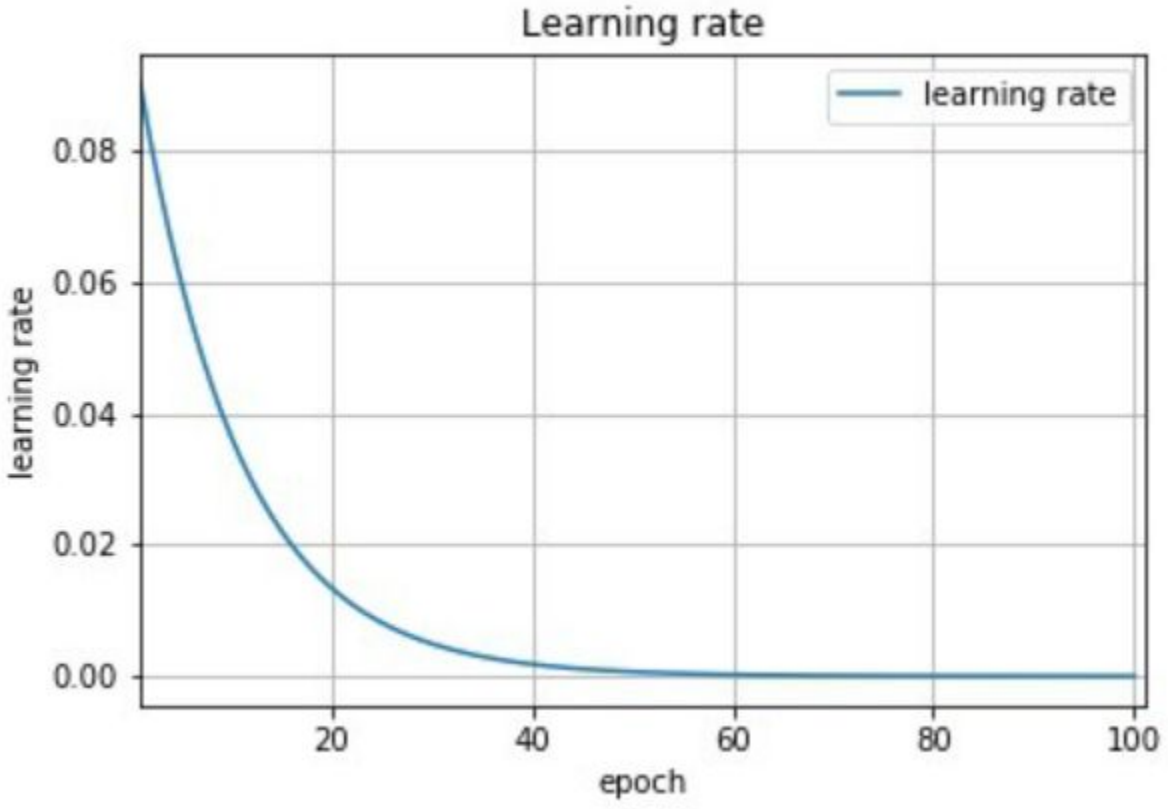
$$lr(\text{epoch}) = lr_0 \lfloor \frac{\text{epoch}}{\text{step}} \rfloor$$



Learning rate schedulers

Exponential decay

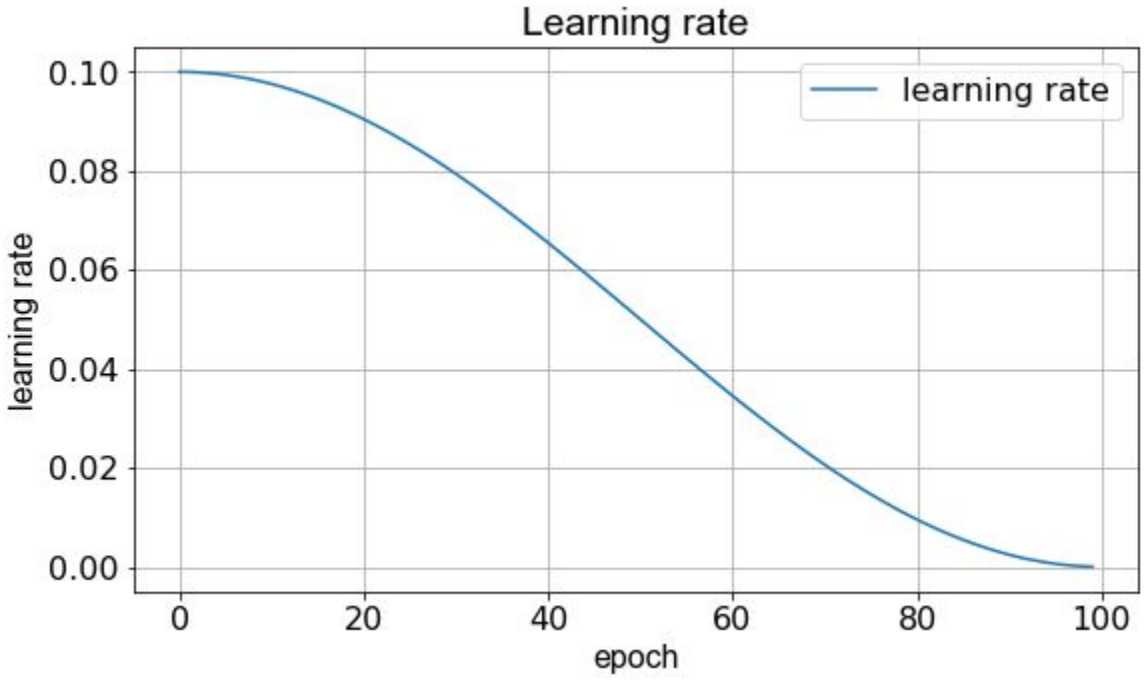
$$lr(\text{epoch}) = lr_0 \cdot e^{-\alpha \cdot \text{epoch}}$$



Learning rate schedulers

Cosine decay

$$lr(\text{epoch}) = \frac{lr_0}{2} \cdot (1 + cos(\frac{\text{epoch} \cdot \pi}{\text{max epoch}}))$$



Learning rate schedulers

Reduce LR on Plateau

- Depends on training/validation set performance
- If loss/accuracy/other metric “plateau” occurs → reduce LR
- Reduction factor is a hyper-parameter

Adaptive optimizers

- AdaGrad / AdaDelta / RMSProp / Adam
- Gradient values are affected by cache values → adaptive LR
- However, common optimizer LR is applied to gradients
- LR scheduler is still needed

Learning rate schedulers

Be careful with them:

- Start with constant LR
- See how model training performs
- Begin adjusting LR only if there is no other way of improving model performance left to try
- Or apply Reduce LR on Plateau

Learning rate warmup

Motivation:

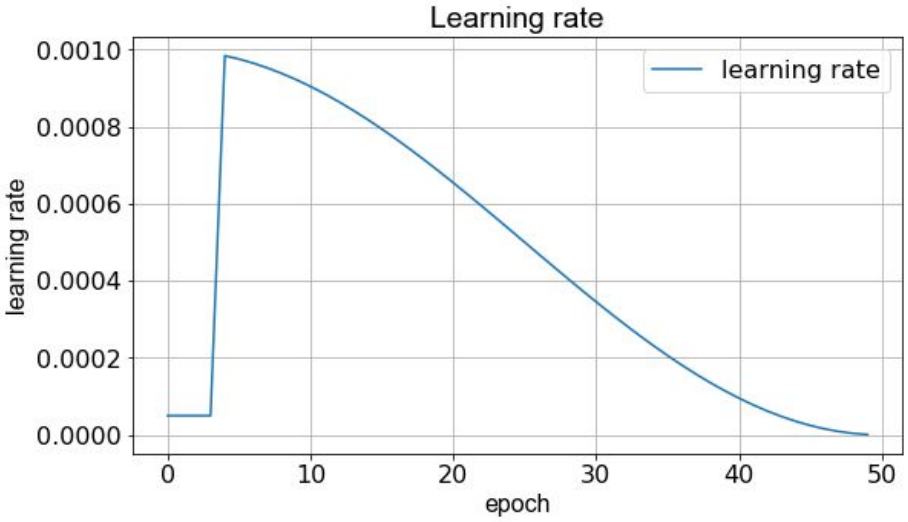
- model weight initialization is random
- initial gradient weight updates are high
- during early GD iterations model can get “early over-fitting”
- several epochs may require to overcome this effect
- (depends on the dataset structure)

source

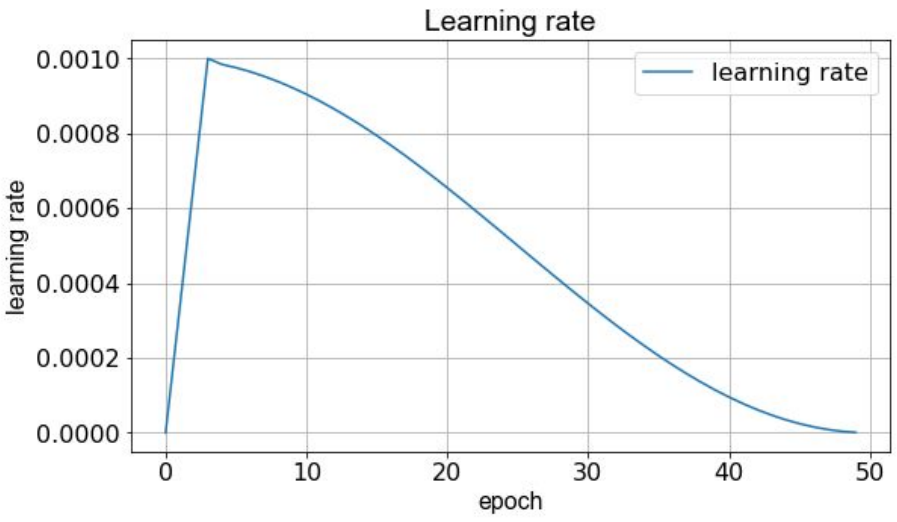
Learning rate warmup

Solution:

- reduce learning rate for several first epochs (warmup period)
- two options:
  - constant reduced learning rate for warm up period
  - gradually increase learning rate from reduced to initial value



(a) constant warm-up



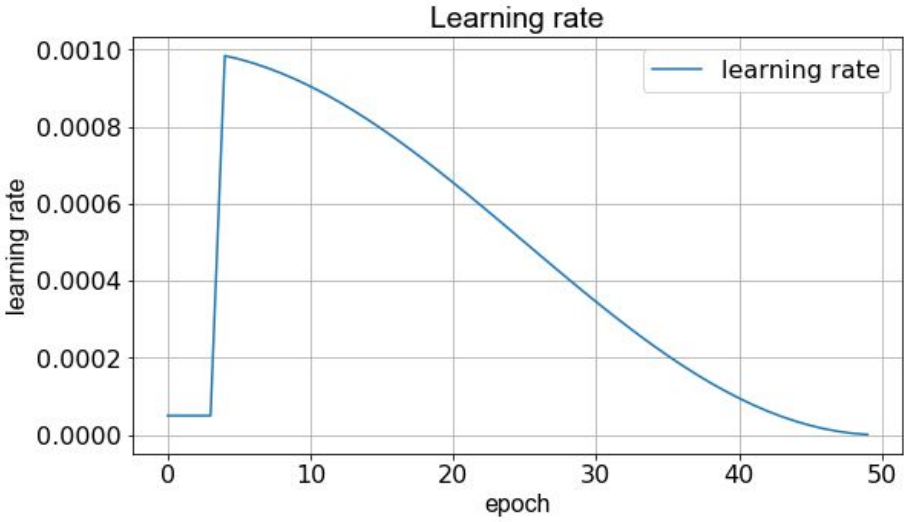
(a) gradual warm-up

source

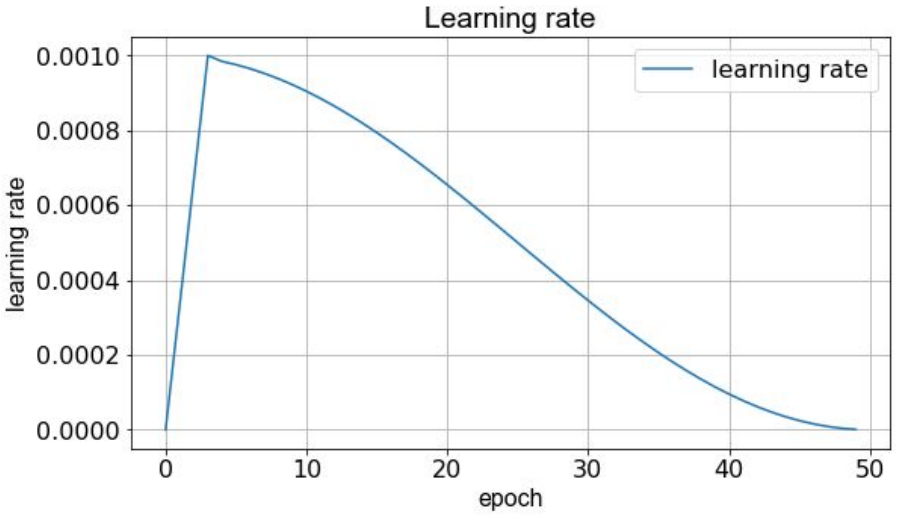
Learning rate warmup

Result:

- speed up model convergence
- small performance boost



(a) constant warm-up



(a) gradual warm-up

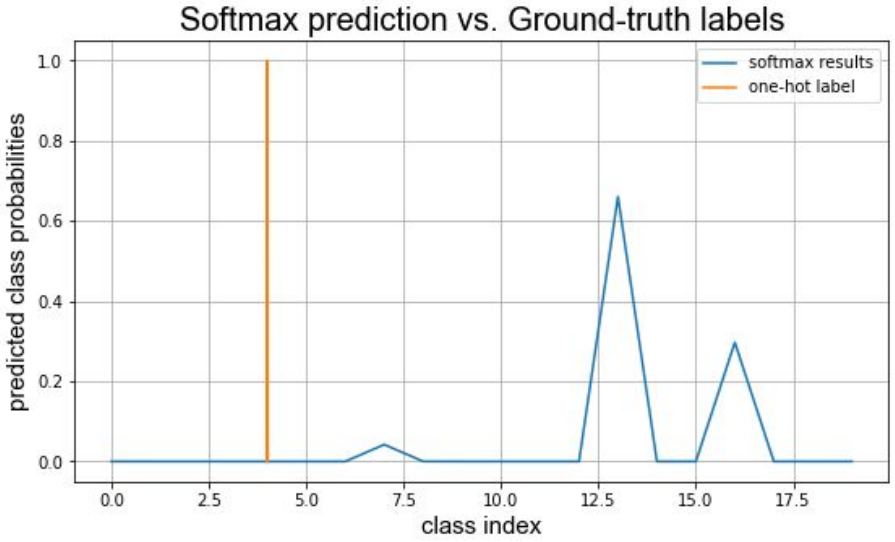
source

Regularization

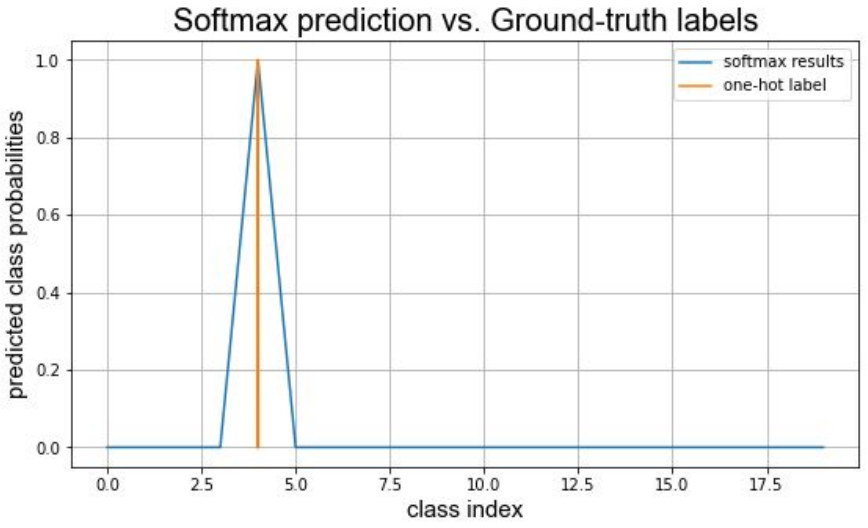
Label Smoothing Regularization

Motivation (for classification):

- cross-entropy loss maximizes log-likelihood between one-hot targets and model predictions
- hard (one-hot) targets → model predicts large logits
- model is overconfident in predictions→ may lead to overfitting



(a) initial epochs



(a) later epochs

source

Label Smoothing Regularization

Motivation (for classification):

- Interpretation in terms of minimized cost function: minimize standard CE with new term

standard CE regularization term

$$H(q', p) = -\sum_{k=1}^{K} \log p(k) q'(k) = (1-\epsilon) H(q, p) + \epsilon H(u, p)$$

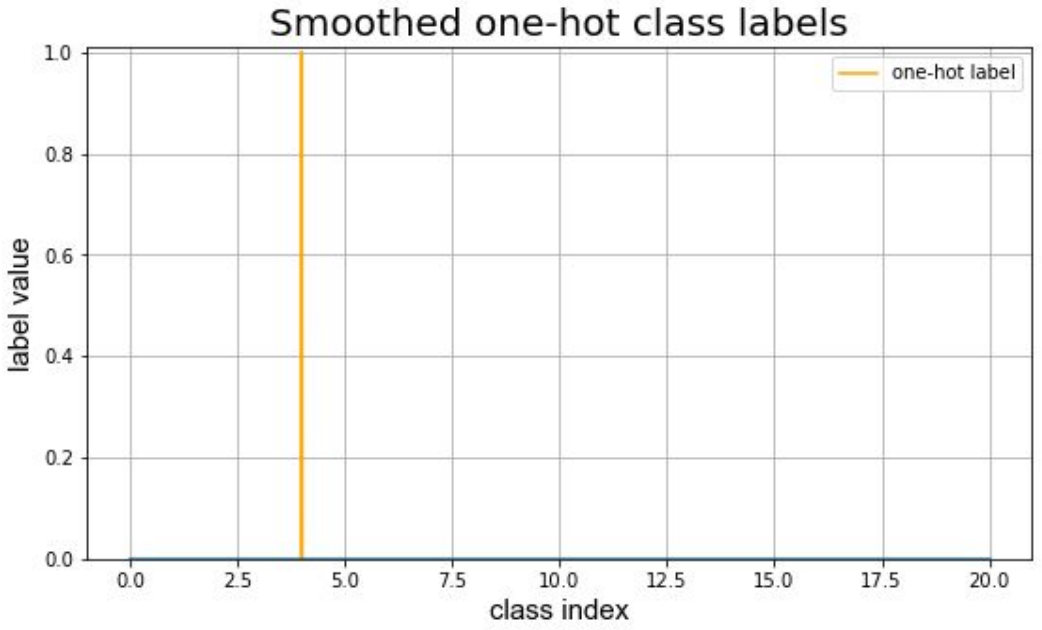
Thus, LSR is equivalent to replacing a single cross-entropy loss $H(q, p)$ with a pair of such losses $H(q, p)$ and $H(u, p)$.

source

Label Smoothing Regularization

- hard one-hot targets:

$$y_k = \begin{cases} 1, & \text{if } k - \text{ground-truth class}, k \in \{1, .., K\} \\ 0, & \text{otherwise} \end{cases}$$

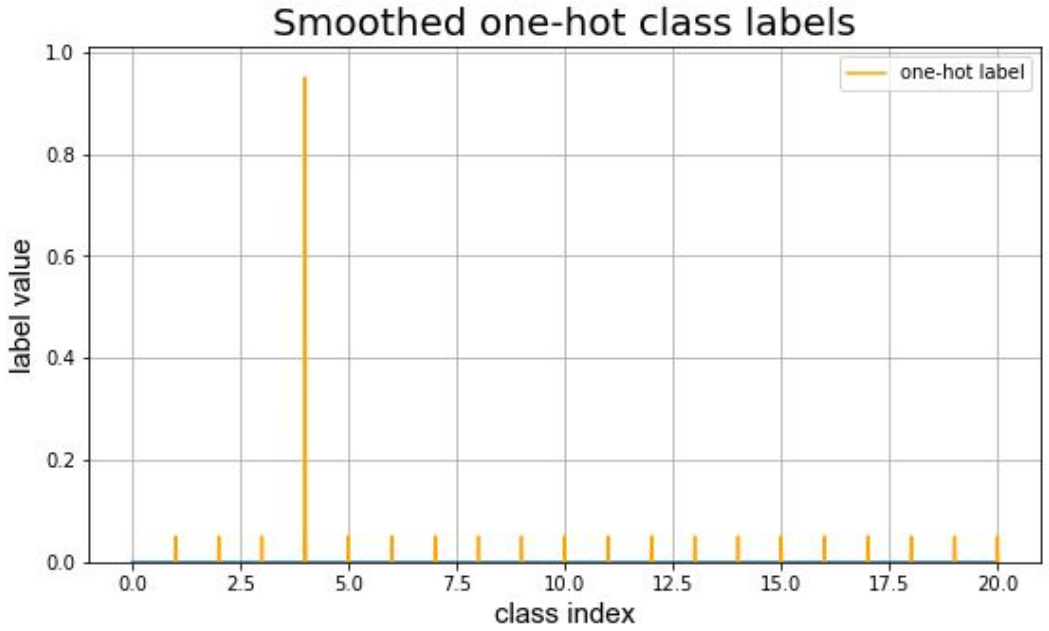


source

Label Smoothing Regularization

- smooth one-hot targets: add a smoothing term **α**: 0 < **α** << 1

$$y_k = \begin{cases} 1-\alpha, & \text{if } k - \text{ground-truth class, } k \in \{1,..,K\} \\ \frac{\alpha}{K}, & \text{otherwise} \end{cases}$$



source

Label Smoothing Regularization

- smooth one-hot targets: add a smoothing term **α**: 0 < **α** << 1

$$y_k = \begin{cases} 1-\alpha, & \text{if } k - \text{ground-truth class}, k \in \{1, .., K\} \\ \frac{\alpha}{K}, & \text{otherwise} \end{cases}$$

- encourages the difference between the logit of the correct class and the logits of incorrect classes to be a constant dependent on **α**
- encourages the activations of the penultimate layer to be close to the template of the correct class and equally distant to the templates of the incorrect classes.

source

Label Smoothing: Results

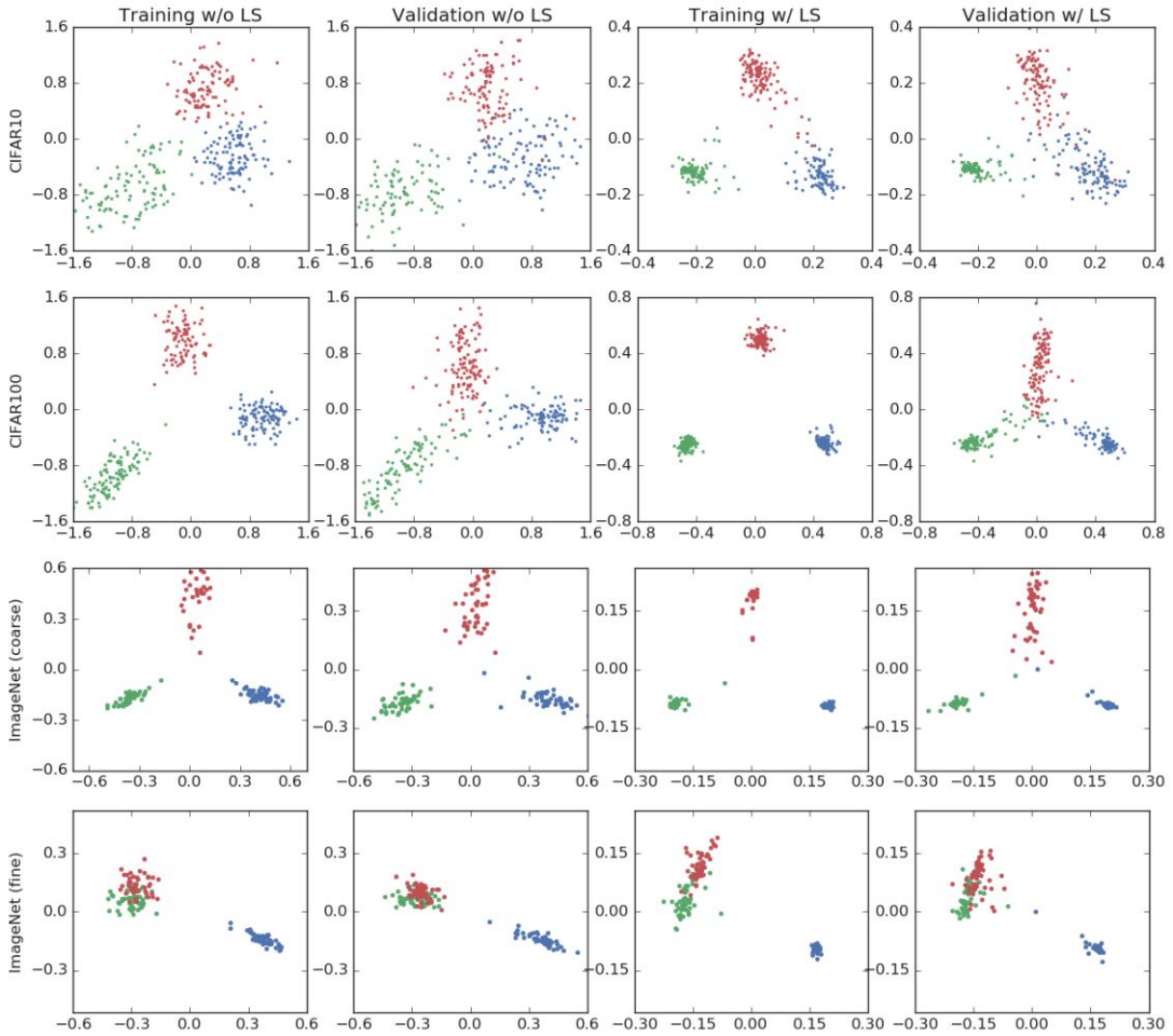


Figure 1: Visualization of penultimate layer's activations of: AlexNet/CIFAR-10 (first row), CIFAR-100/ResNet-56 (second row) and ImageNet/Inception-v4 with three semantically different classes (third row) and two semantically similar classes plus a third one (fourth row).

source

Mixup

- Idea: mix features and labels with linear interpolation

Contribution Motivated by these issues, we introduce a simple and data-agnostic data augmentation routine, termed *mixup* (Section 2). In a nutshell, *mixup* constructs virtual training examples

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

(x_i, y_i) and (x_j, y_j) are two examples drawn at random from our training data, and $\lambda \in [0, 1]$. Therefore, *mixup* extends the training distribution by incorporating the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated targets. *mixup* can be implemented in a few lines of code, and introduces minimal computation overhead.

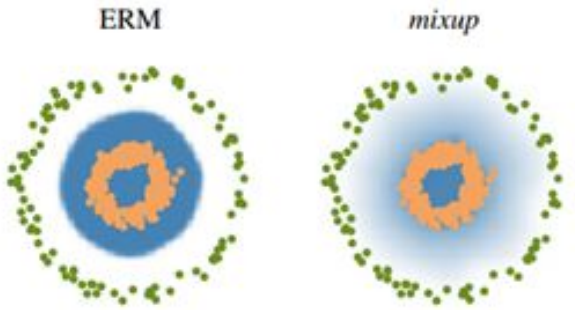
source

Mixup

- Extremely strong regularizer
- A form data augmentation,
- According to authors learned manifold is smoother
- (see toy model example)

```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of *mixup* training in PyTorch.



(b) Effect of *mixup* ($\alpha = 1$) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates $p(y = 1|x)$.

source

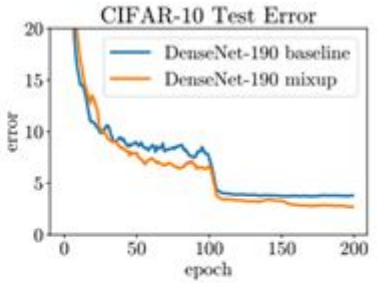
Mixup: Results

Model	Method	Epochs	Top-1 Error	Top-5 Error
ResNet-50	ERM (Goyal et al., 2017)	90	23.5	-
	mixup $\alpha = 0.2$	90	**23.3**	**6.6**
ResNet-101	ERM (Goyal et al., 2017)	90	22.1	-
	mixup $\alpha = 0.2$	90	**21.5**	**5.6**
ResNeXt-101 32*4d	ERM (Xie et al., 2016)	100	21.2	-
	ERM	90	21.2	5.6
	mixup $\alpha = 0.4$	90	**20.7**	**5.3**
ResNeXt-101 64*4d	ERM (Xie et al., 2016)	100	20.4	5.3
	mixup $\alpha = 0.4$	90	**19.8**	**4.9**
ResNet-50	ERM	200	23.6	7.0
	mixup $\alpha = 0.2$	200	**22.1**	**6.1**
ResNet-101	ERM	200	22.0	6.1
	mixup $\alpha = 0.2$	200	**20.8**	**5.4**
ResNeXt-101 32*4d	ERM	200	21.3	5.9
	mixup $\alpha = 0.4$	200	**20.1**	**5.0**

Table 1: Validation errors for ERM and *mixup* on the development set of ImageNet-2012.

Dataset	Model	ERM	*mixup*
CIFAR-10	PreAct ResNet-18	5.6	**4.2**
	WideResNet-28-10	3.8	**2.7**
	DenseNet-BC-190	3.7	**2.7**
CIFAR-100	PreAct ResNet-18	25.6	**21.1**
	WideResNet-28-10	19.4	**17.5**
	DenseNet-BC-190	19.0	**16.8**

(a) Test errors for the CIFAR experiments.



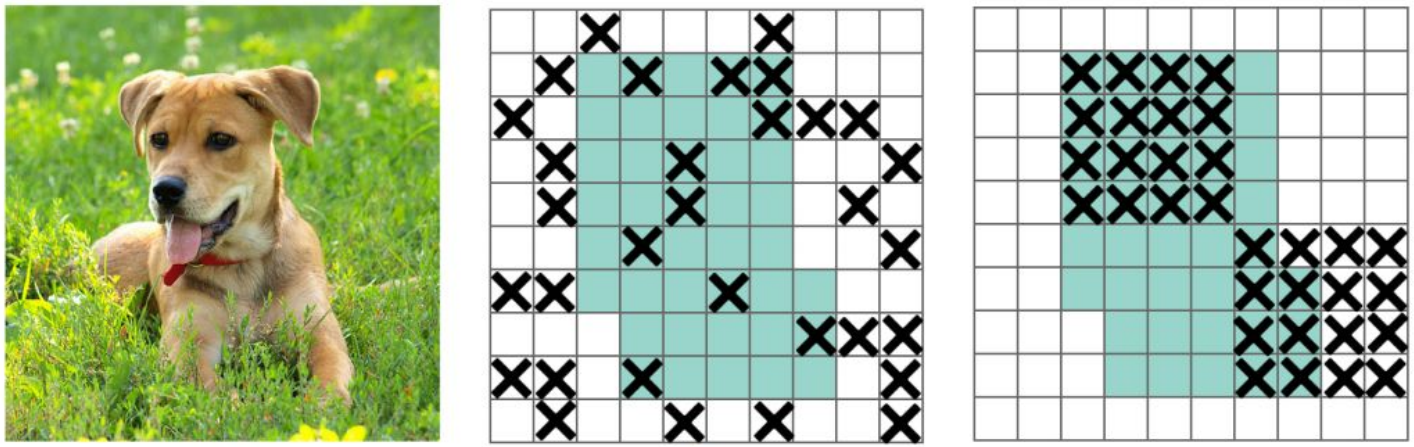
(b) Test error evolution for the best ERM and *mixup* models.

Figure 3: Test errors for ERM and *mixup* on the CIFAR experiments.

source

DropBlock

- Much like dropout, but for intermediate layers
- Zero-out continuous activations tensor regions
- Designed for spatially connected features
- Get 1-2% more performance from your model



(a) (b) (c)

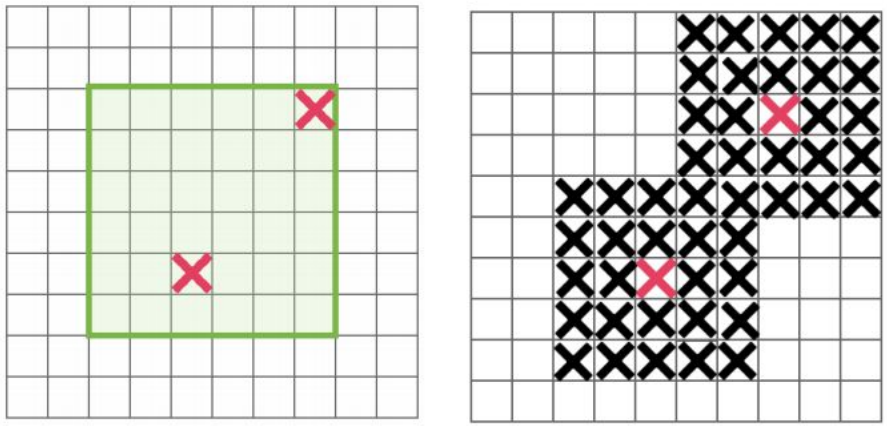
Figure 1: (a) input image to a convolutional neural network. The green regions in (b) and (c) include the activation units which contain semantic information in the input image. Dropping out activations at random is not effective in removing semantic information because nearby activations contain closely related information. Instead, dropping continuous regions can remove certain semantic information (e.g., head or feet) and consequently enforcing remaining units to learn features for classifying input image.

source

DropBlock

Algorithm 1 DropBlock

Input: output activations of a layer (A), $block\_size$, γ, $mode$
if $mode$ == *Inference* **then**
    return A
end if
Randomly sample mask M: $M_{i,j} \sim Bernoulli(\gamma)$
For each zero position $M_{i,j}$, create a spatial square mask with the center being $M_{i,j}$, the width, height being $block\_size$ and set all the values of M in the square to be zero (see Figure 2).
Apply the mask: $A = A \times M$
Normalize the features: $A = A \times \textbf{count}(M)/\textbf{count\_ones}(M)$



(a) (b)

Figure 2: Mask sampling in DropBlock. (a) On every feature map, similar to dropout, we first sample a mask M. We only sample mask from shaded green region in which each sampled entry can expanded to a mask fully contained inside the feature map. (b) Every zero entry on M is expanded to $block\_size \times block\_size$ zero block.

source

DropBlock: Results

Model	top-1(%)	top-5(%)
ResNet-50	76.51 ± 0.07	93.20 ± 0.05
ResNet-50 + dropout (kp=0.7) [1]	76.80 ± 0.04	93.41 ± 0.04
ResNet-50 + DropPath (kp=0.9) [17]	77.10 ± 0.08	93.50 ± 0.05
ResNet-50 + SpatialDropout (kp=0.9) [20]	77.41 ± 0.04	93.74 ± 0.02
ResNet-50 + Cutout [23]	76.52 ± 0.07	93.21 ± 0.04
ResNet-50 + AutoAugment [27]	77.63	93.82
ResNet-50 + label smoothing (0.1) [28]	77.17 ±0.05	93.45 ±0.03
ResNet-50 + DropBlock, (kp=0.9)	78.13 ± 0.05	94.02 ± 0.02
ResNet-50 + DropBlock (kp=0.9) + label smoothing (0.1)	78.35 ± 0.05	94.15 ± 0.03

Table 1: Summary of validation accuracy on ImageNet dataset for ResNet-50 architecture. For dropout, DropPath, and SpatialDropout, we trained models with different $keep\_prob$ values and reported the best result. DropBlock is applied with $block\_size = 7$. We report average over 3 runs.

source

DropBlock

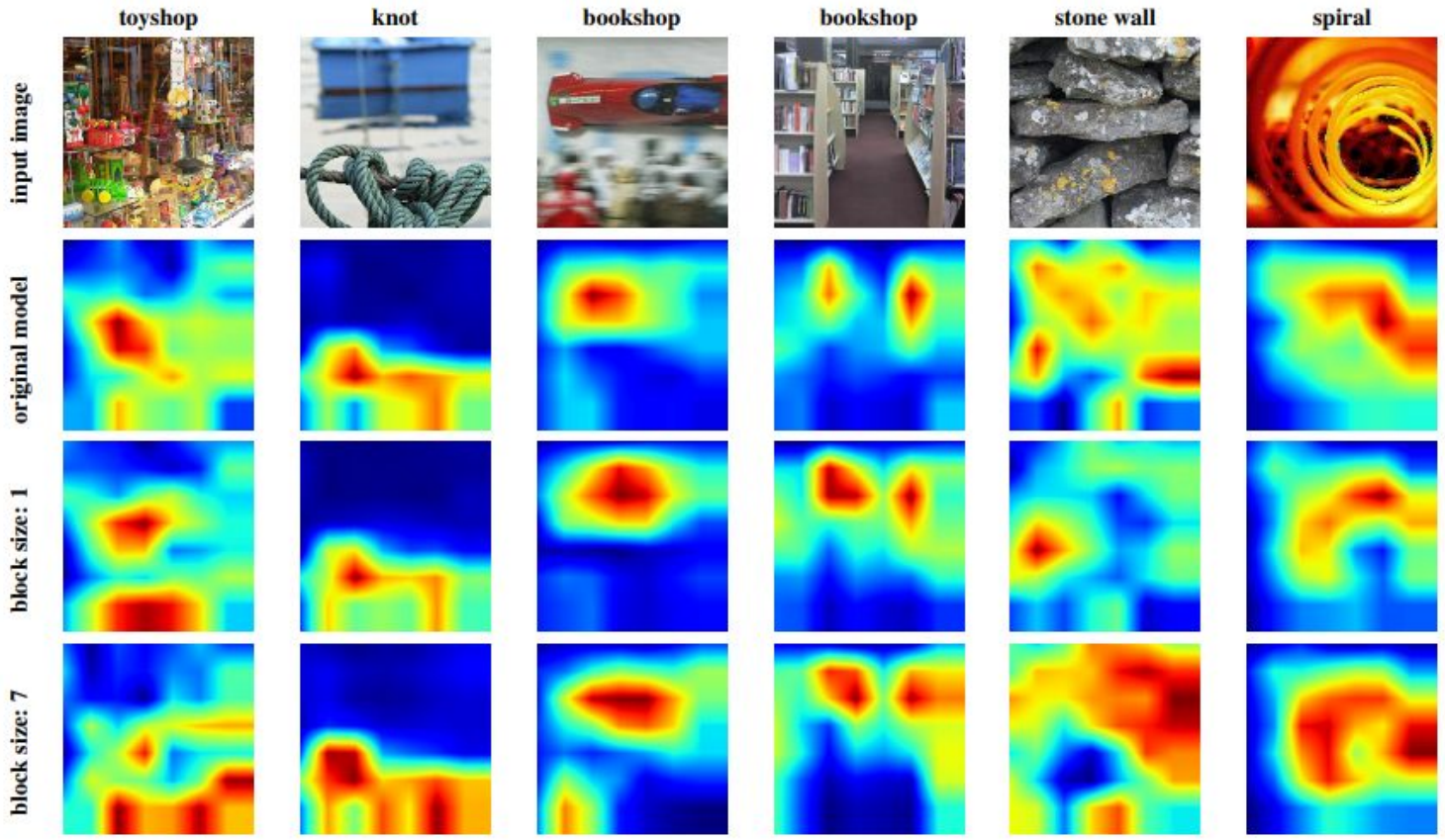


Figure 6: Class activation mapping (CAM) [29] for ResNet-50 model trained without DropBlock and trained with DropBlock with the *block_size* of 1 or 7. The model trained with DropBlock tends to focus on several spatially distributed regions.

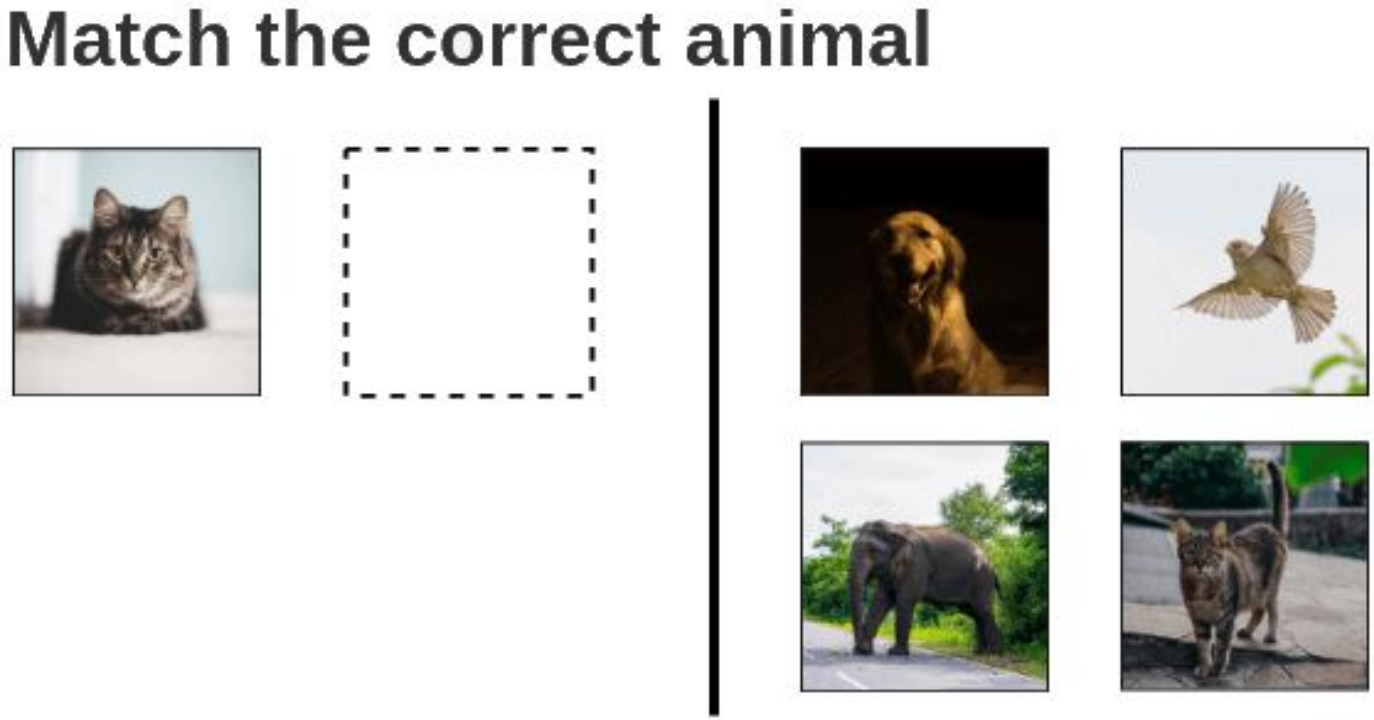
CAM

source

Thanks for watching!

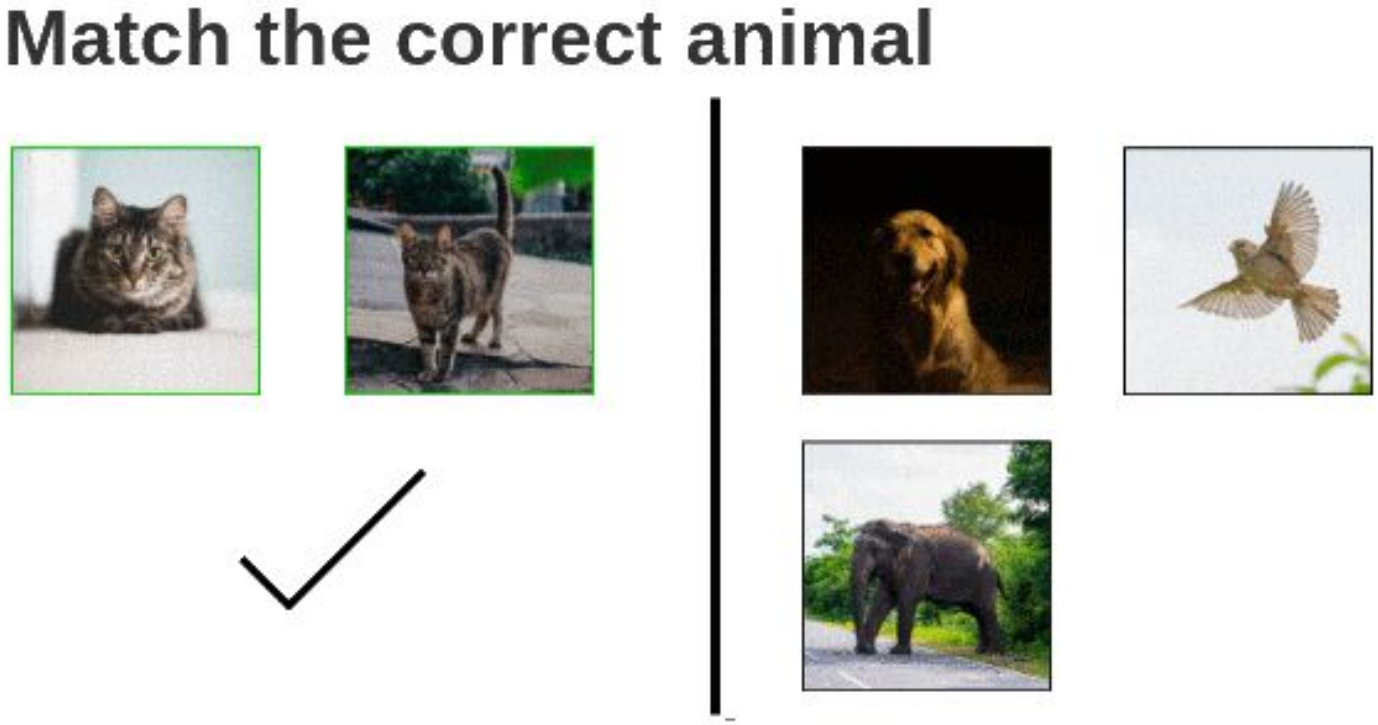
(optional) Contrastive Learning

Contrastive Learning Concept



source and paper

Contrastive Learning Concept



source and paper

Contrastive Learning Concept

- Learn similar/dissimilar representations from data
- Data has to be organized into similar/dissimilar pairs
- NN becomes feature extractor
- Fine-tune it for target task in supervised manner

source

Contrastive Learning Concept

We need the following:

1. Labeled similar/dissimilar data
2. Image-to-representation mapper
3. Similarity estimation algorithm

source

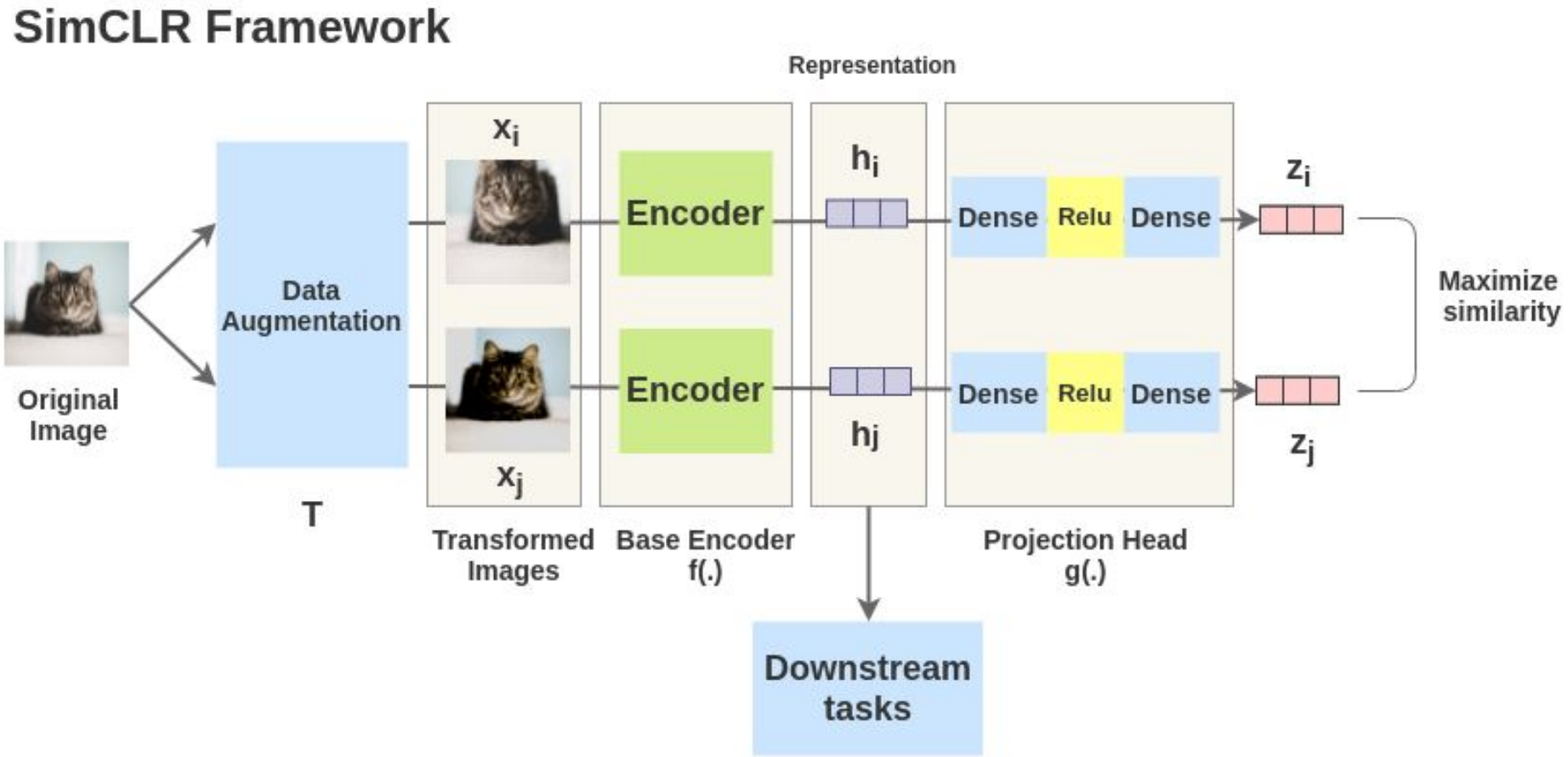
Contrastive Learning Concept

We need the following:

1. Labeled similar/dissimilar data → actually optional
2. Image-to-representation mapper → CNN encoder
3. Similarity estimation algorithm → cosine similarity

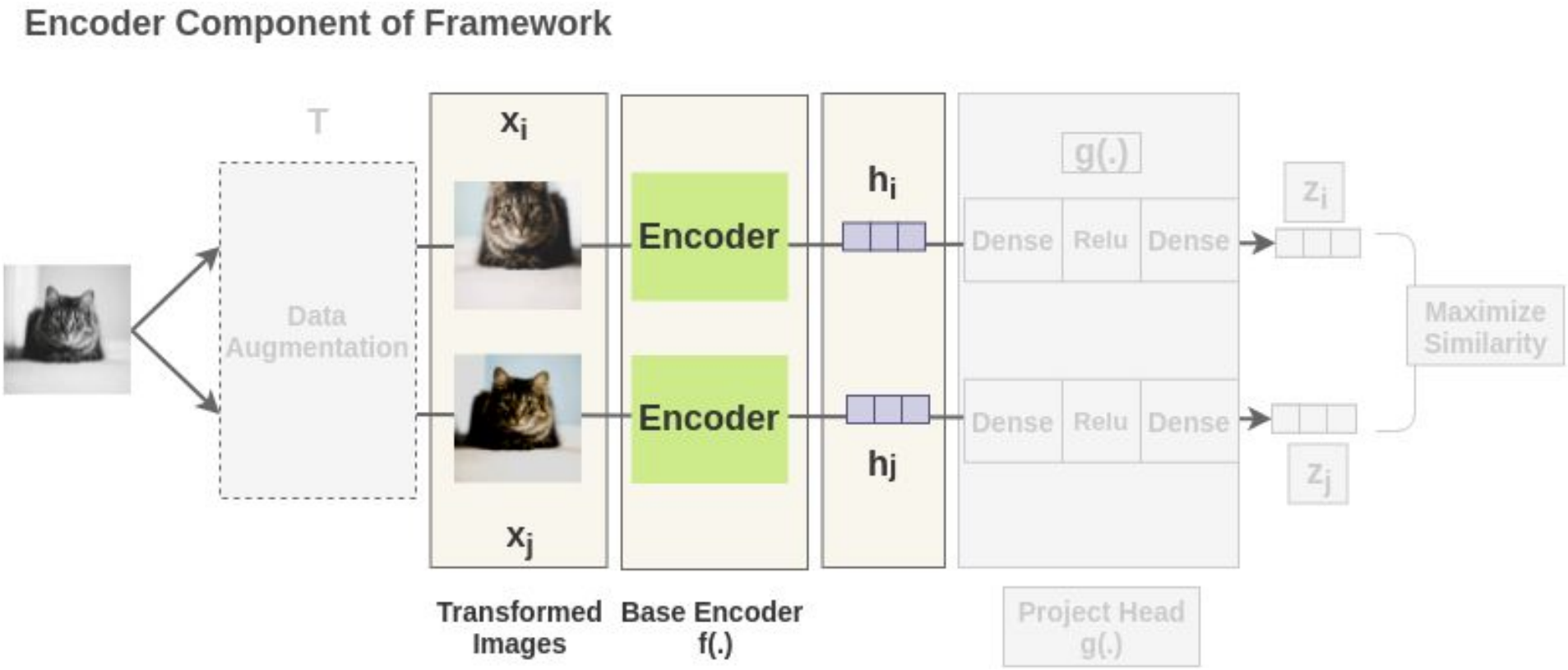
source

Contrastive Learning Concept



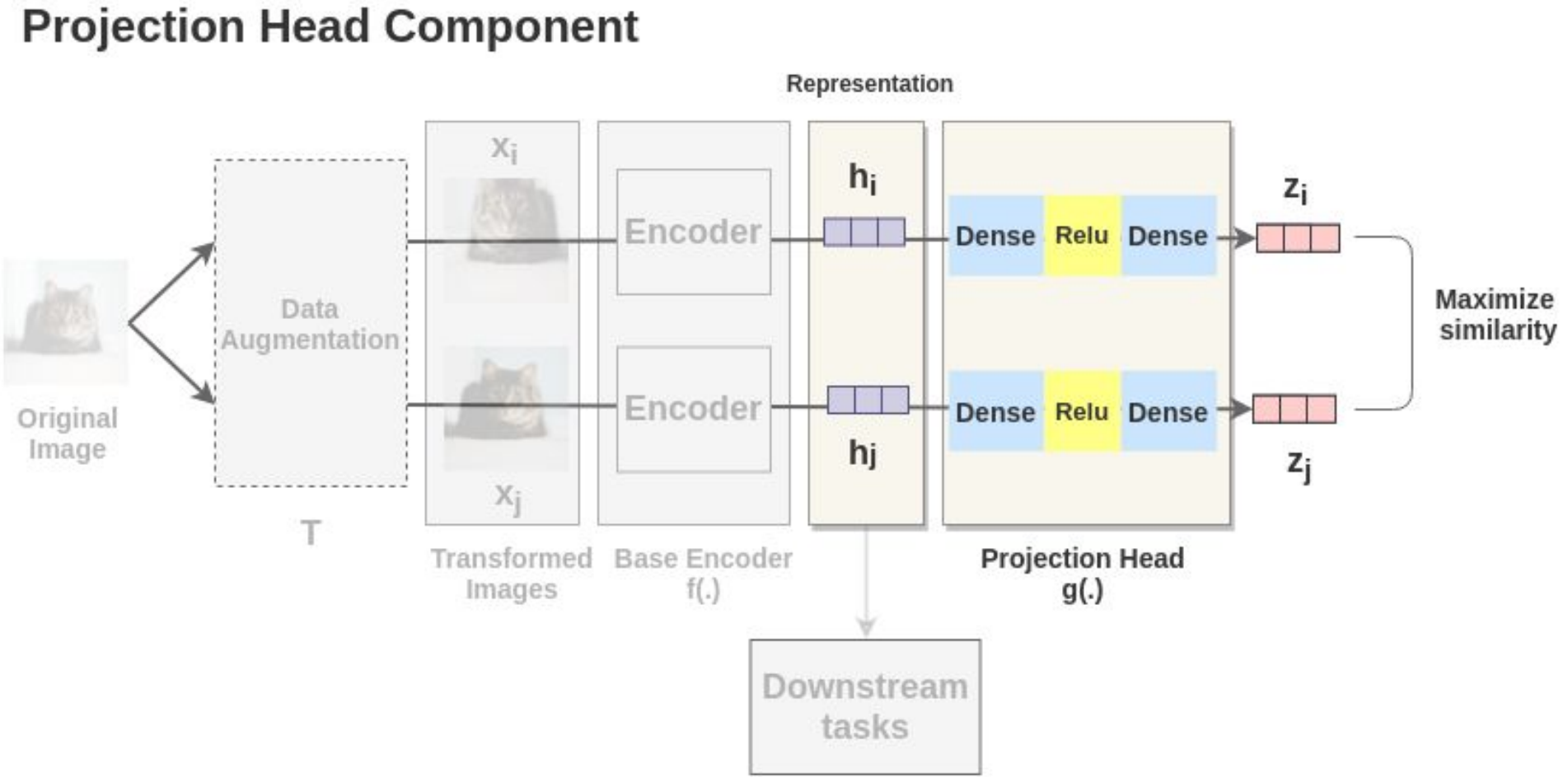
source and paper

Contrastive Learning Concept



source and paper

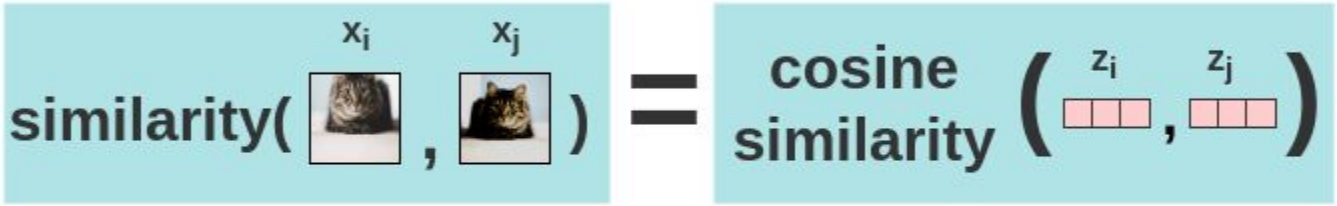
Contrastive Learning Concept



source and paper

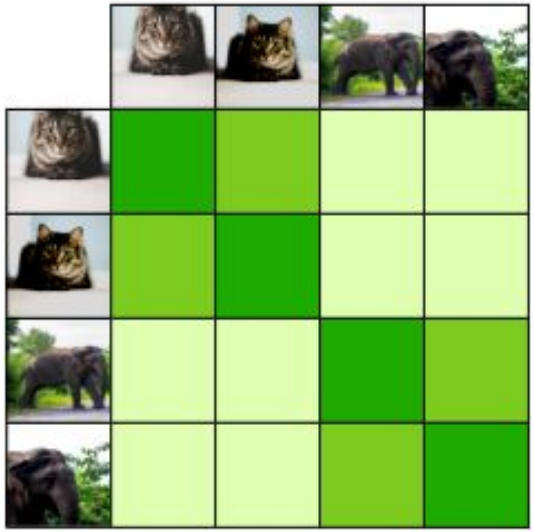
Contrastive Learning Concept

Similarity Calculation of Augmented Images



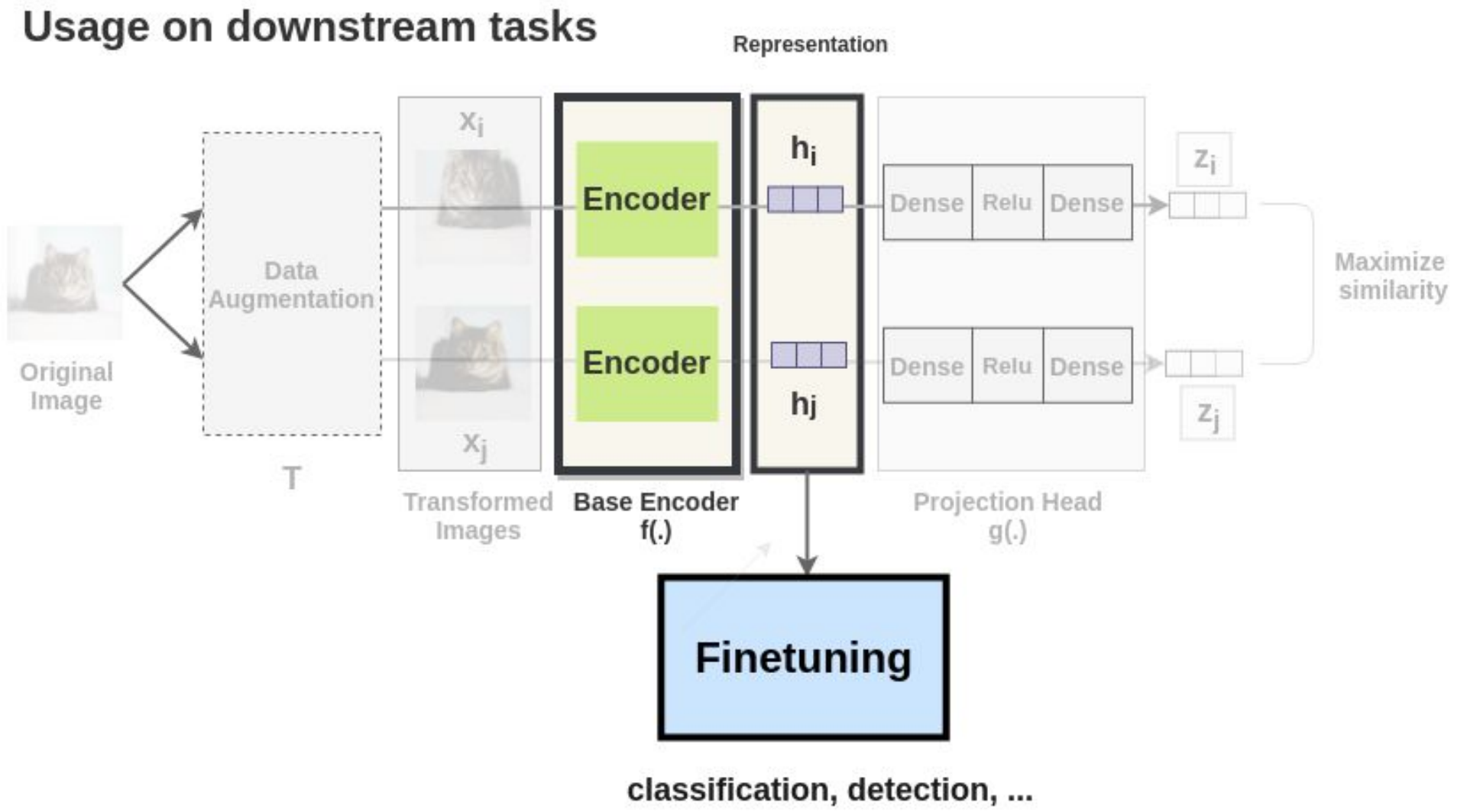
$$s_{i,j} = \frac{z_i^T z_j}{(\tau ||z_i|| ||z_j||)}$$

Pairwise cosine similarity



[source](#) and [paper](#)

Contrastive Learning Concept



source and paper

Thanks for watching!