



UNIVERSITY OF TECHNOLOGY
IN THE EUROPEAN CAPITAL OF CULTURE
CHEMNITZ

Neurocomputing

Contrastive learning

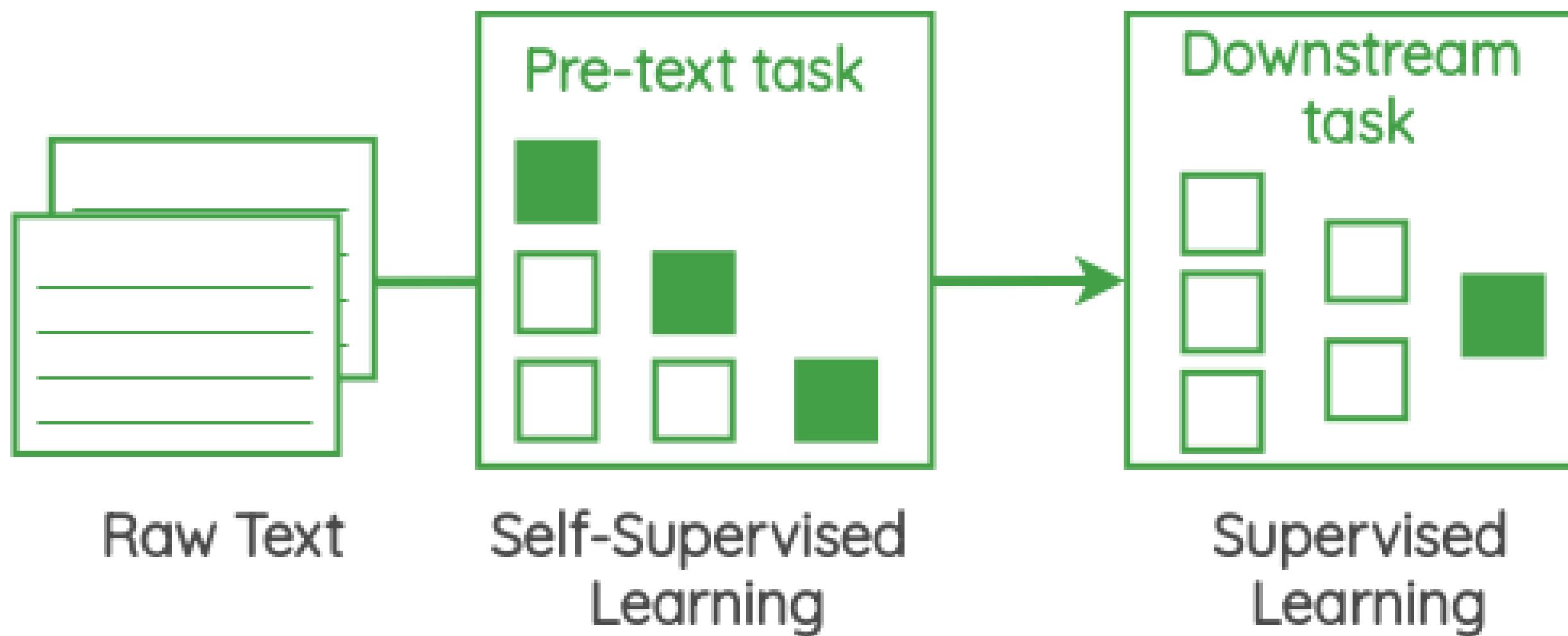
Julien Vitay

Professur für Künstliche Intelligenz - Fakultät für Informatik

1 - Self-supervised learning

Self-supervised learning

- **Supervised learning** uses a supervisory signal (e.g. human-annotated labels) to train a model (classification, regression).
- **Unsupervised learning** only relies on analysing the properties of the data (clustering, dimensionality reduction).
- **Semi-supervised learning** first extract features on raw data (e.g. autoencoder) and then fine-tunes a model on annotated data.
- **Self-supervised learning** creates its own supervisory signal from the raw data to extract features using a **pretext task** or **auxiliary task**. These features can then be used to learn a supervised learning problem (downstream task).

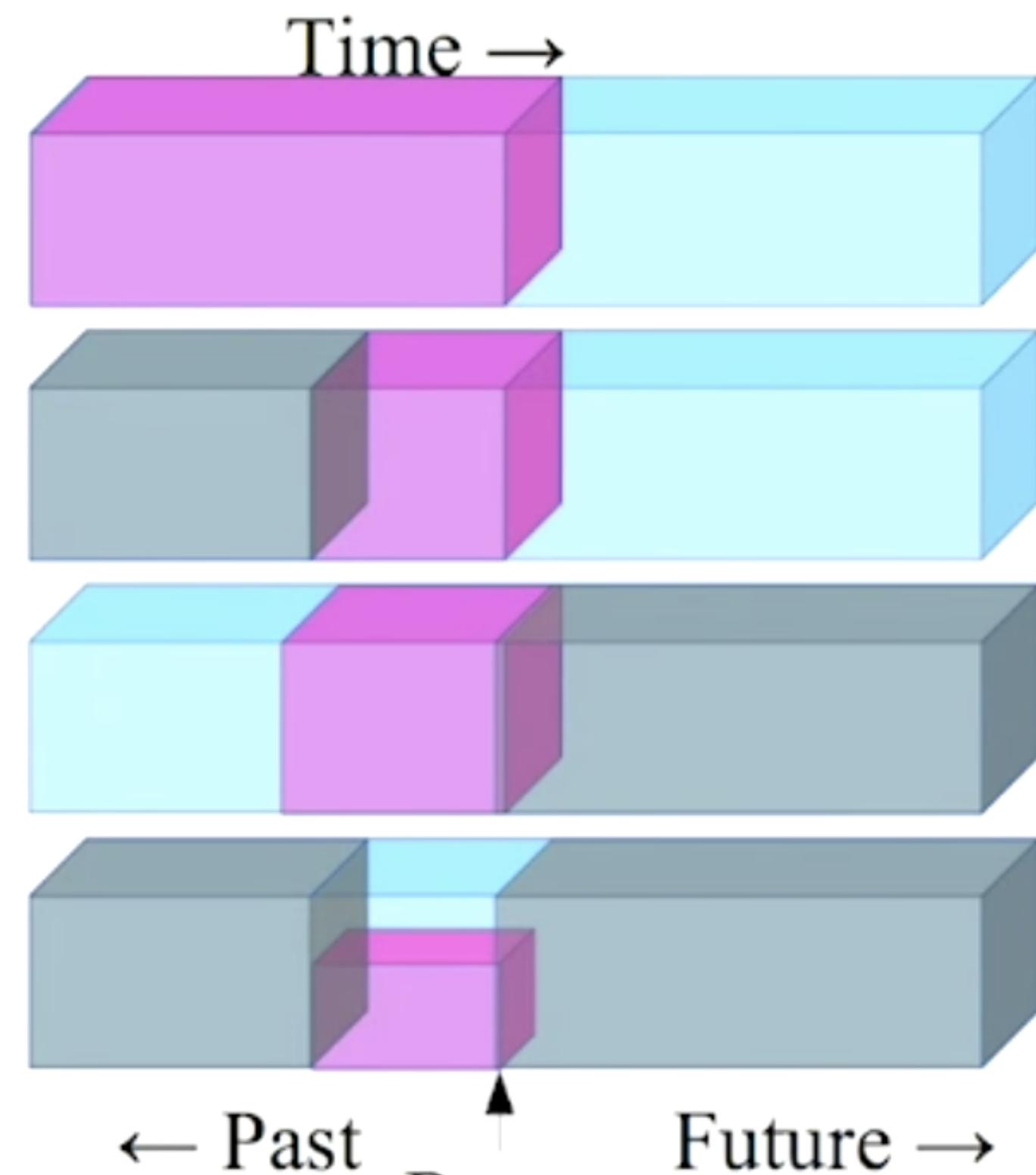


Source: <https://amitness.com/2020/05/self-supervised-learning-nlp/>

Self-supervised learning

- Pretext tasks can be easily and automatically derived from the existing data, such as predicting the future of a signal.

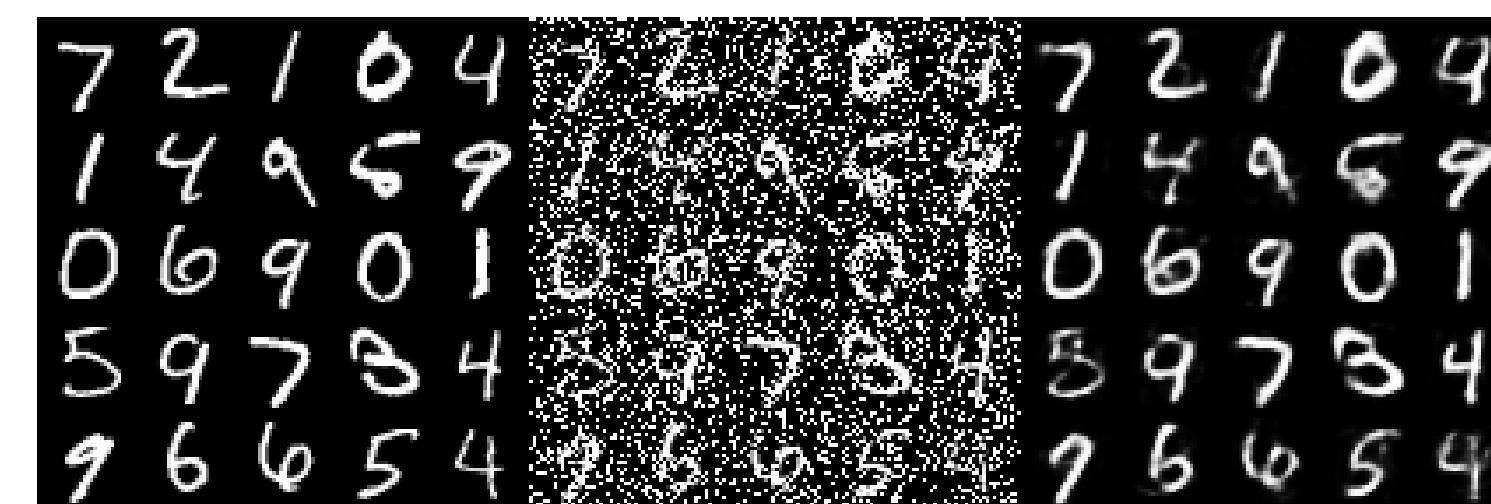
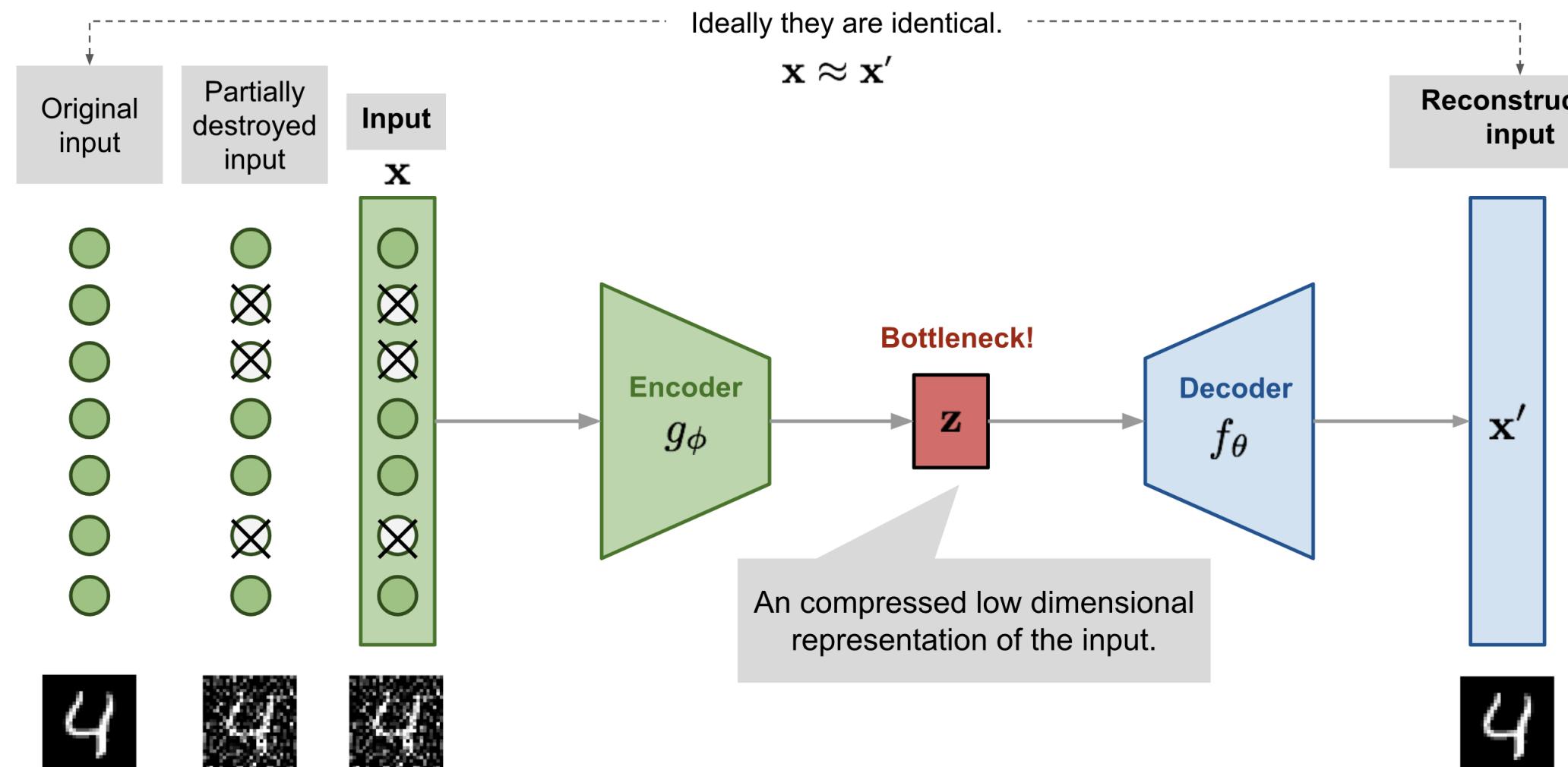
- ▶ Predict any part of the input from any other part.
- ▶ Predict the **future** from the **past**.
- ▶ Predict the **future** from the **recent past**.
- ▶ Predict the **past** from the **present**.
- ▶ Predict the **top** from the **bottom**.
- ▶ Predict the **occluded** from the **visible**
- ▶ Pretend there is a part of the input you don't know and predict that.



Slide: LeCun

Generative models

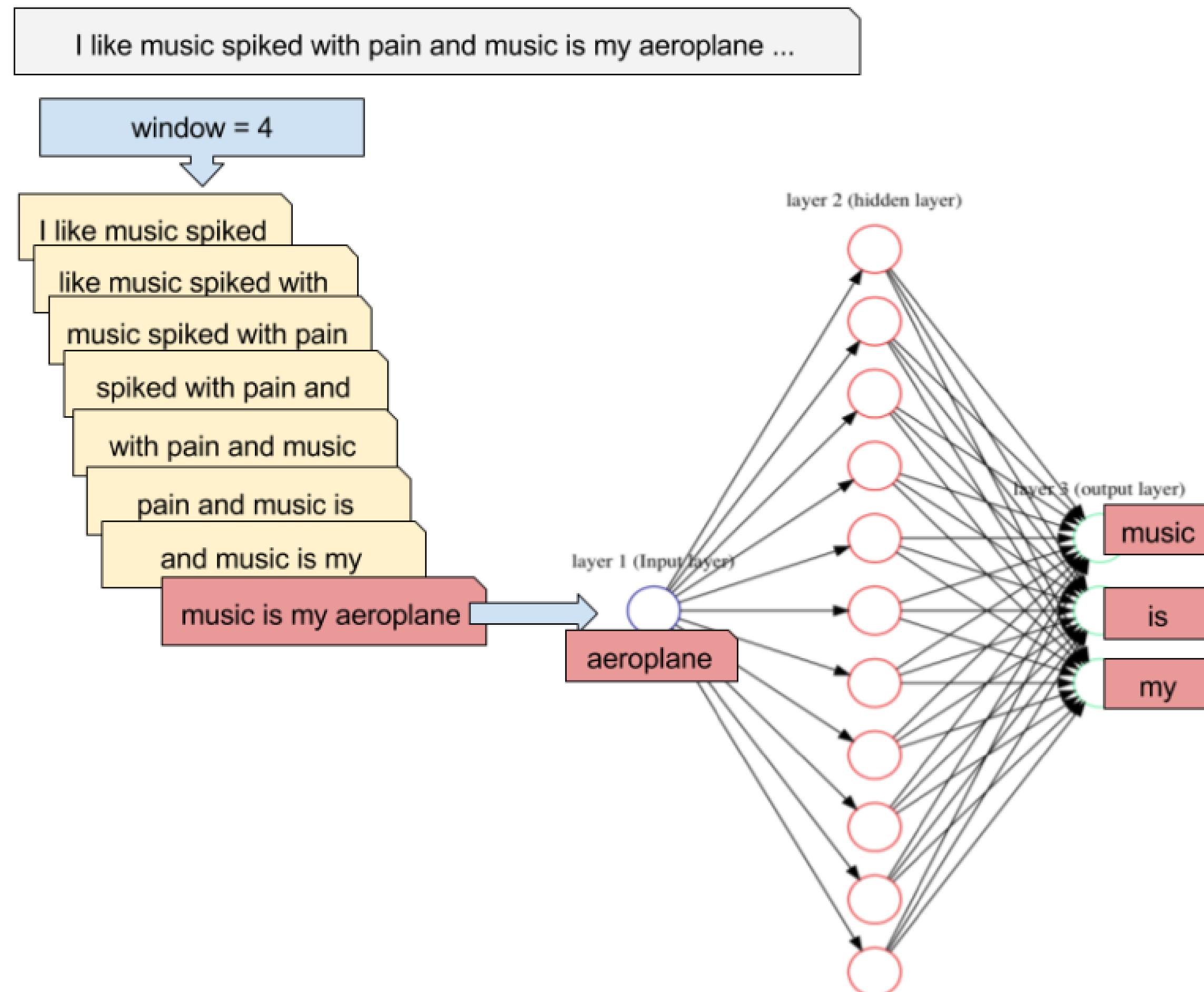
- Generative models (AE, GAN) are somehow self-supervised: reconstructing an image is just a pretext to learn a good latent representation, or to learn to remove noise (denoising AE).



Source : <https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html>

word2vec

- word2vec is trained using the pretext task of predicting the surrounding words in a sentence.



Source: <https://jaxenter.com/deep-learning-search-word2vec-147782.html>

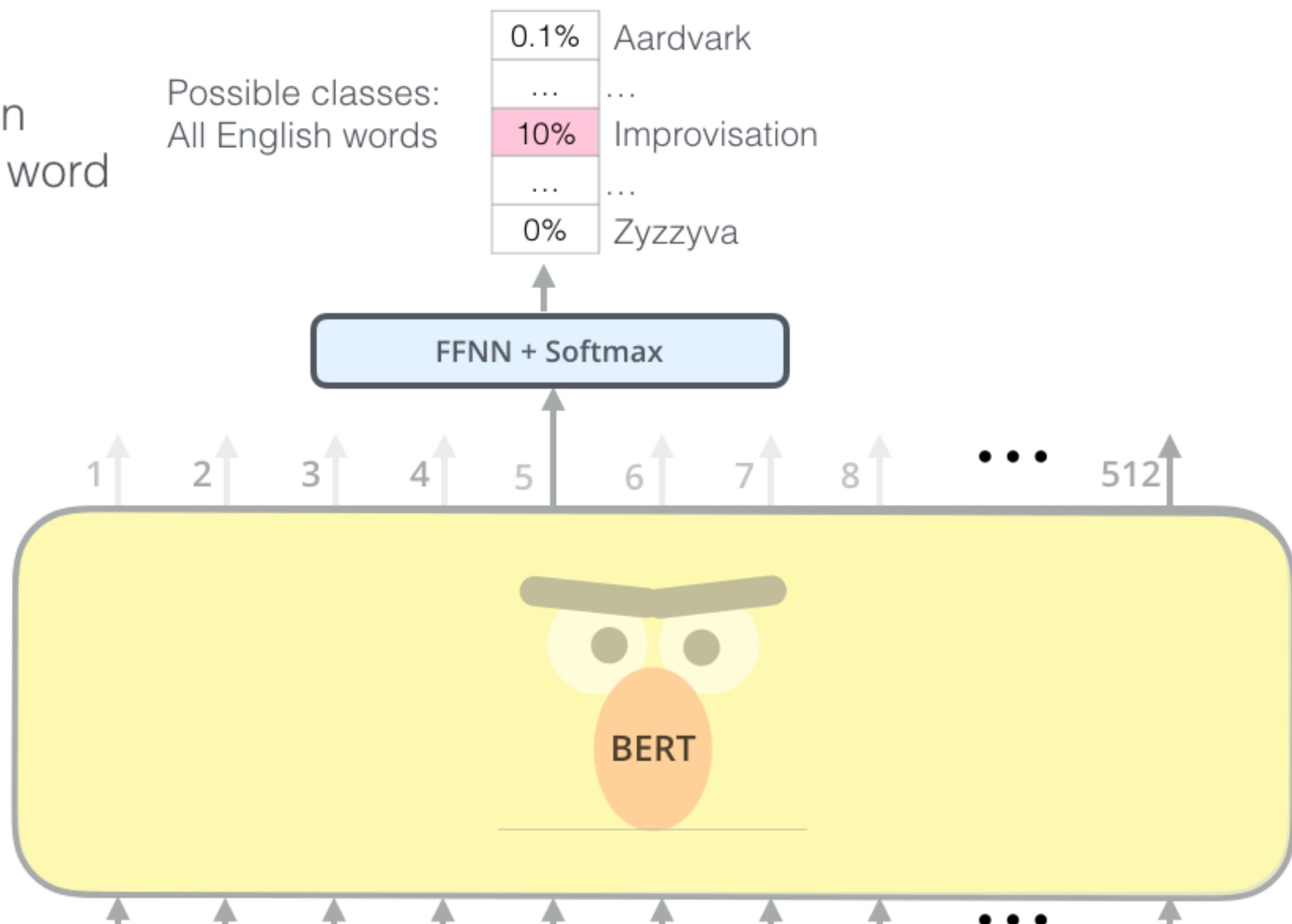
Masked word prediction

Use the output of the masked word's position to predict the masked word

Randomly mask 15% of tokens

Input

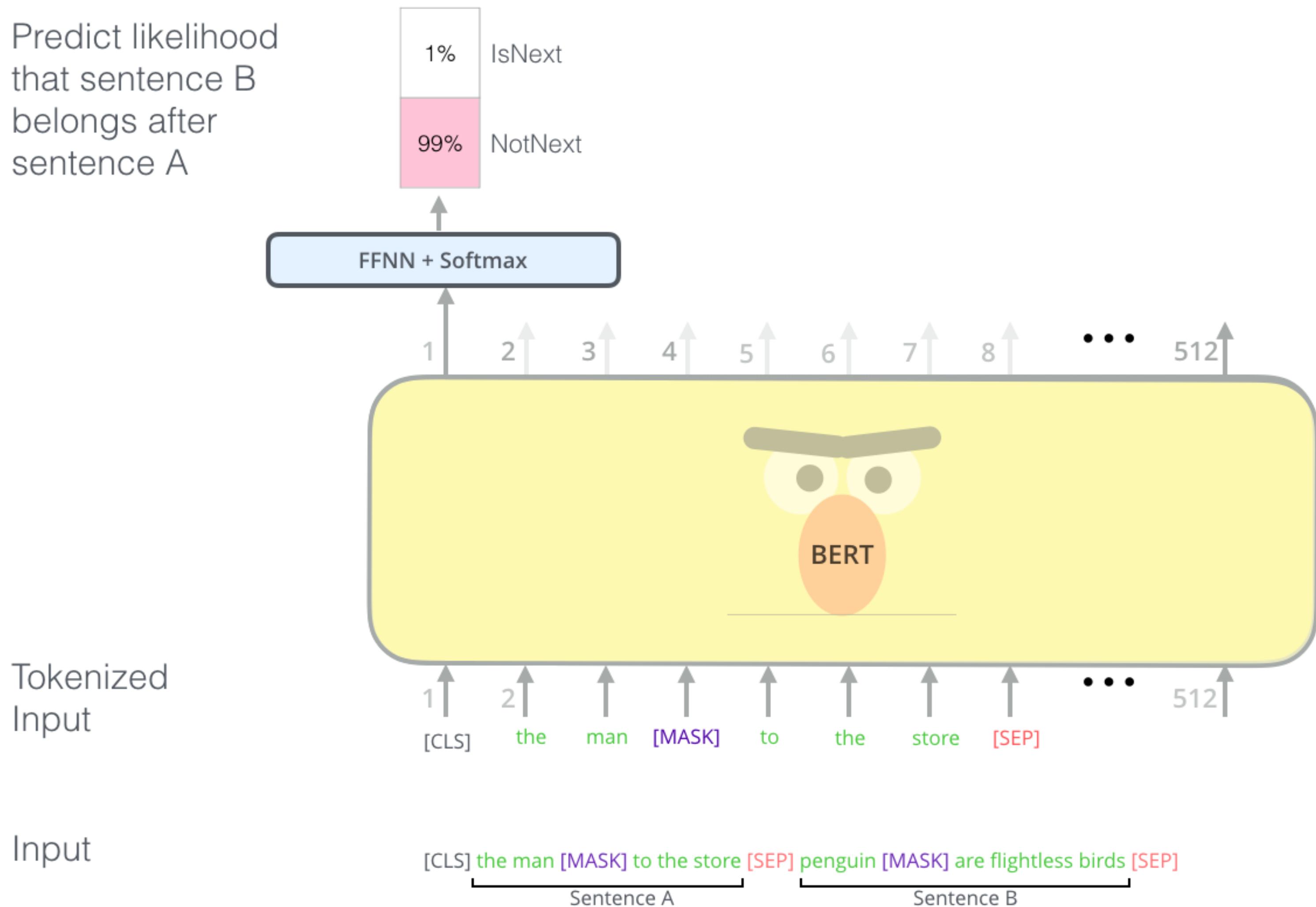
[CLS] ↑ Let's ↑ stick ↑ to ↑ improvement ↑ in ↑ this ↑ skit ↑



Source: <https://jalammar.github.io/illustrated-bert/>

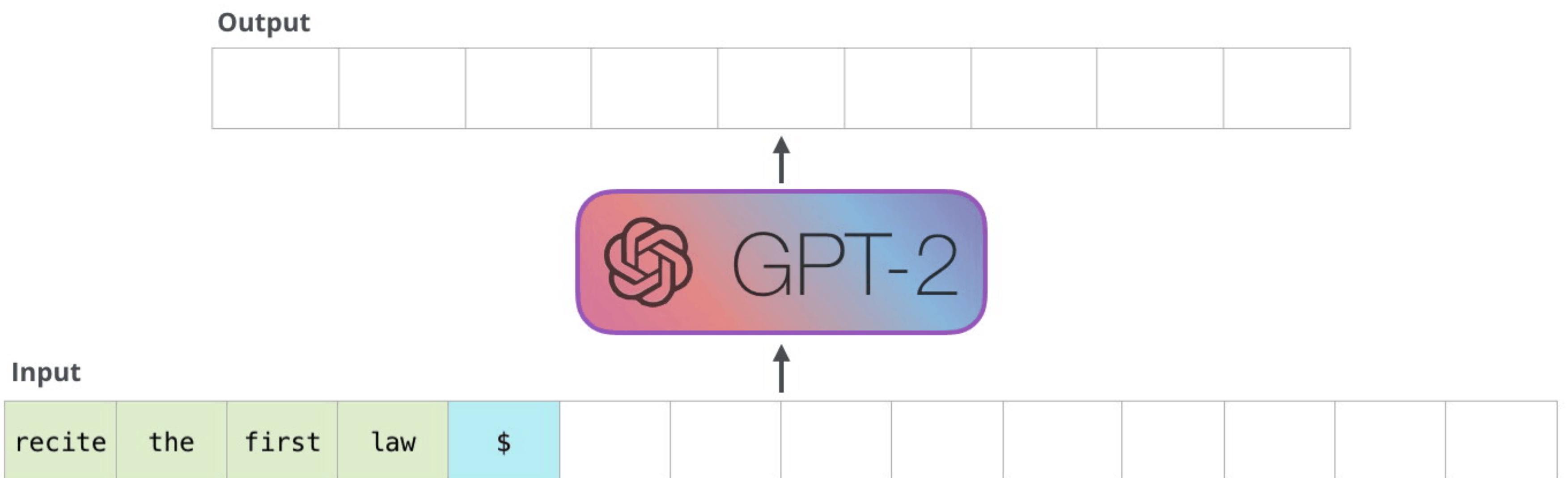
Next sentence prediction

Predict likelihood
that sentence B
belongs after
sentence A



Source: <https://jalammar.github.io/illustrated-bert/>

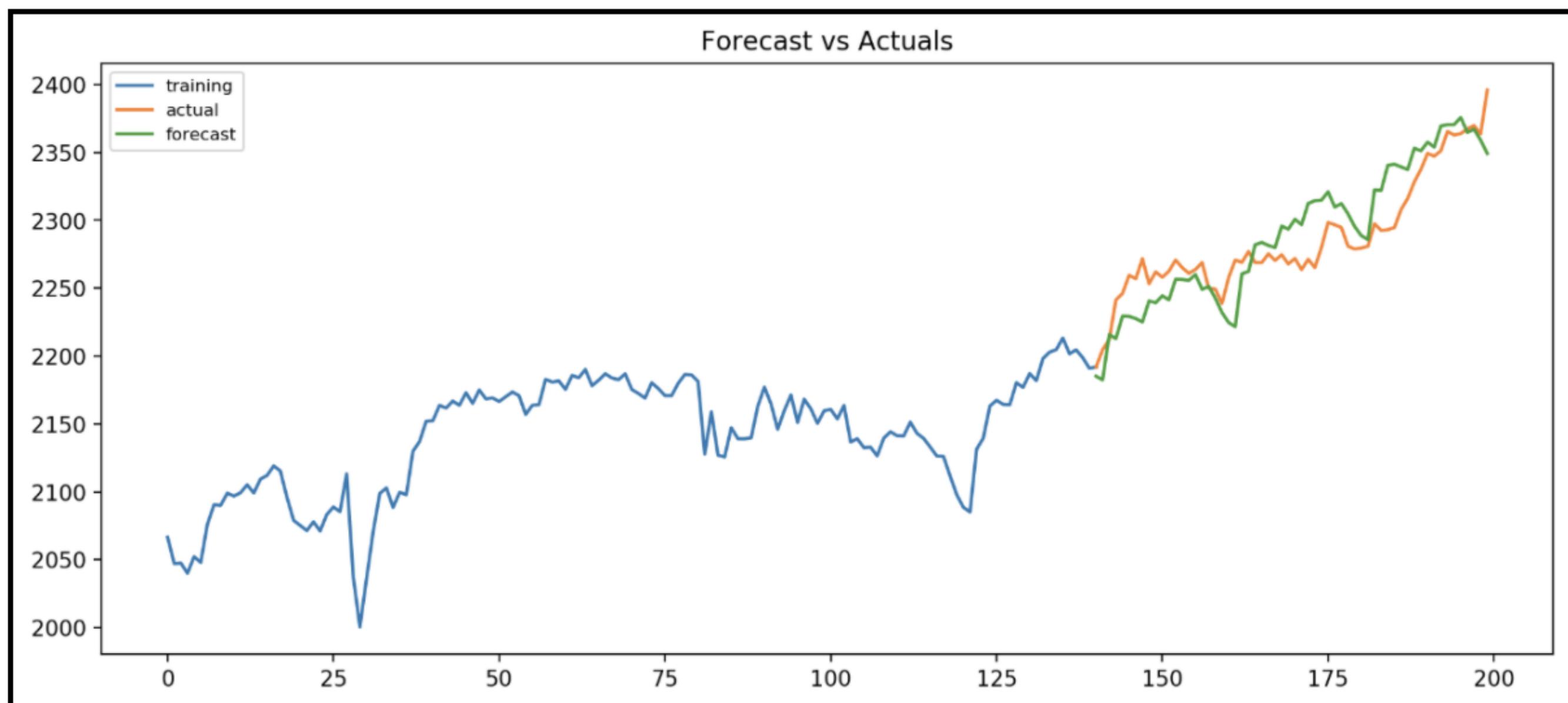
Next word prediction



Source: <https://jalammar.github.io/illustrated-gpt2/>

Autoregression

- Autoregressive models (RNN) are trained to predict the next value \mathbf{x}_{t+1} of a (multivariate) signal based on its history $(\mathbf{x}_{t-T}, \dots, \mathbf{x}_t)$.
- At inference time, they can “unroll” the future by considering their prediction as the future “ground truth”.
- Useful for forecasting: weather, share values, predictive maintenance, etc.



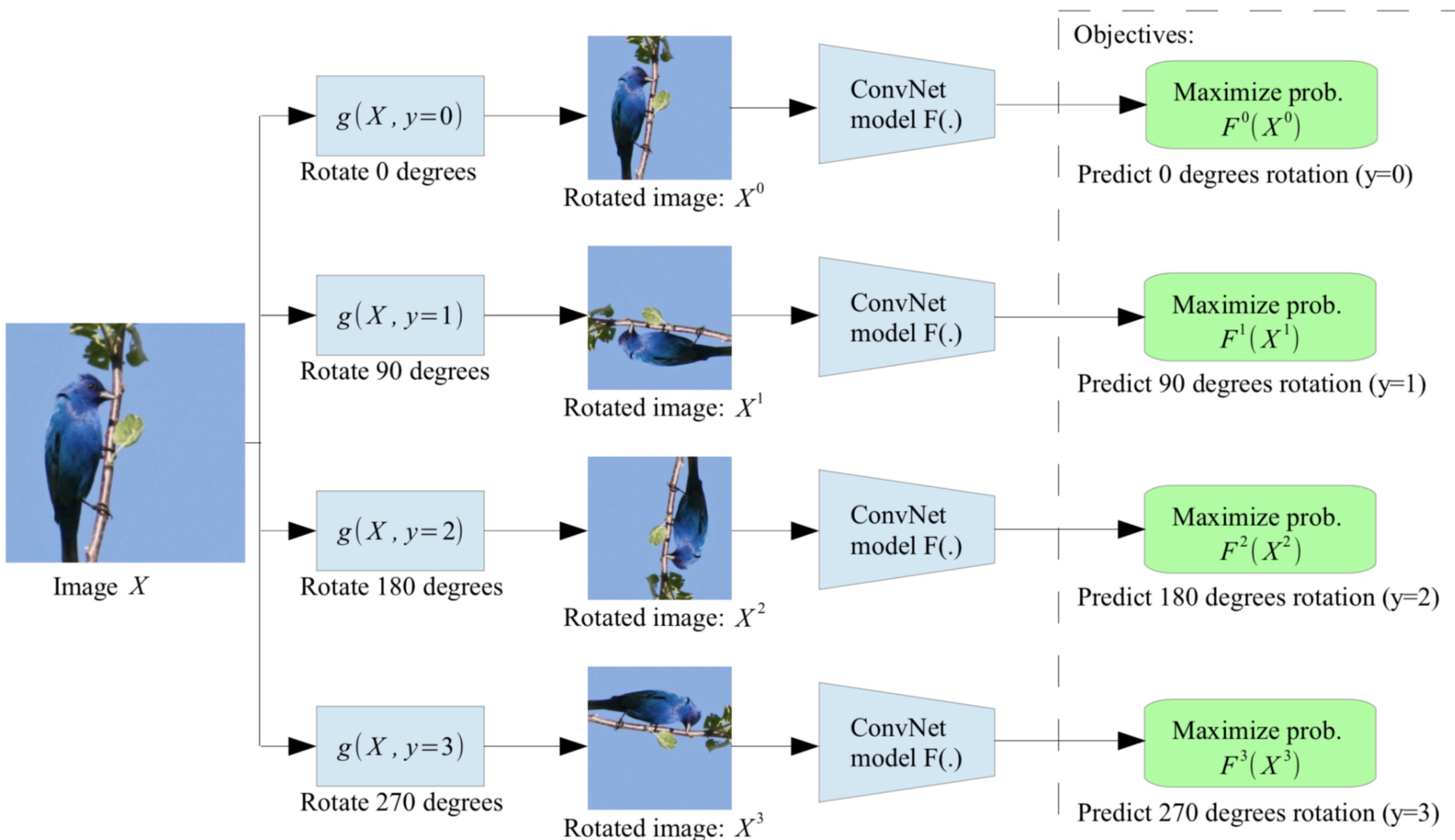
Test RMSE: 23.580

Test Percentage Error: 0.010%

Source: <https://saas.berkeley.edu/rp/arima>

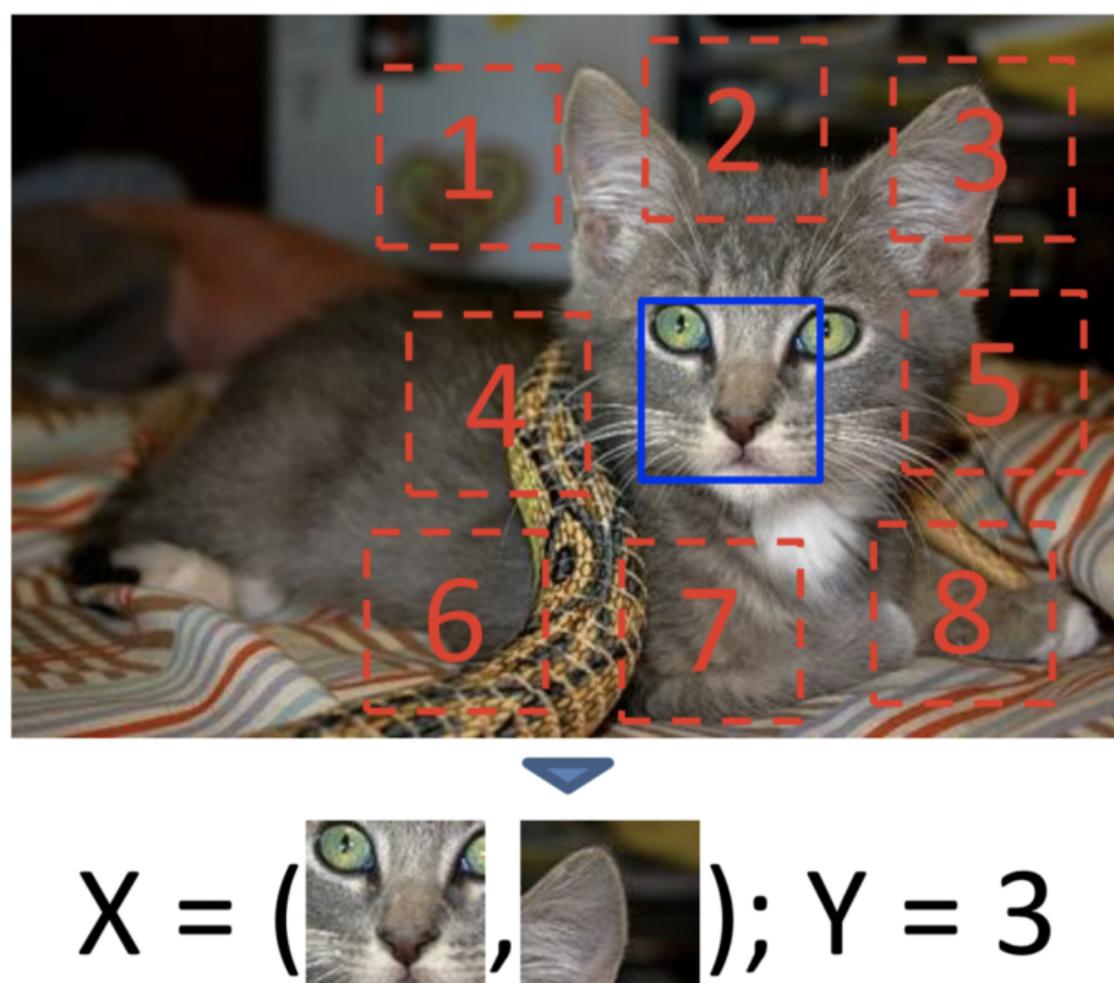
Rotation prediction

- Rotations are applied to an image and the CNN has to guess which one has been applied.
- By doing so, it has to learn visual features that “understand” what the regular position of an object is.

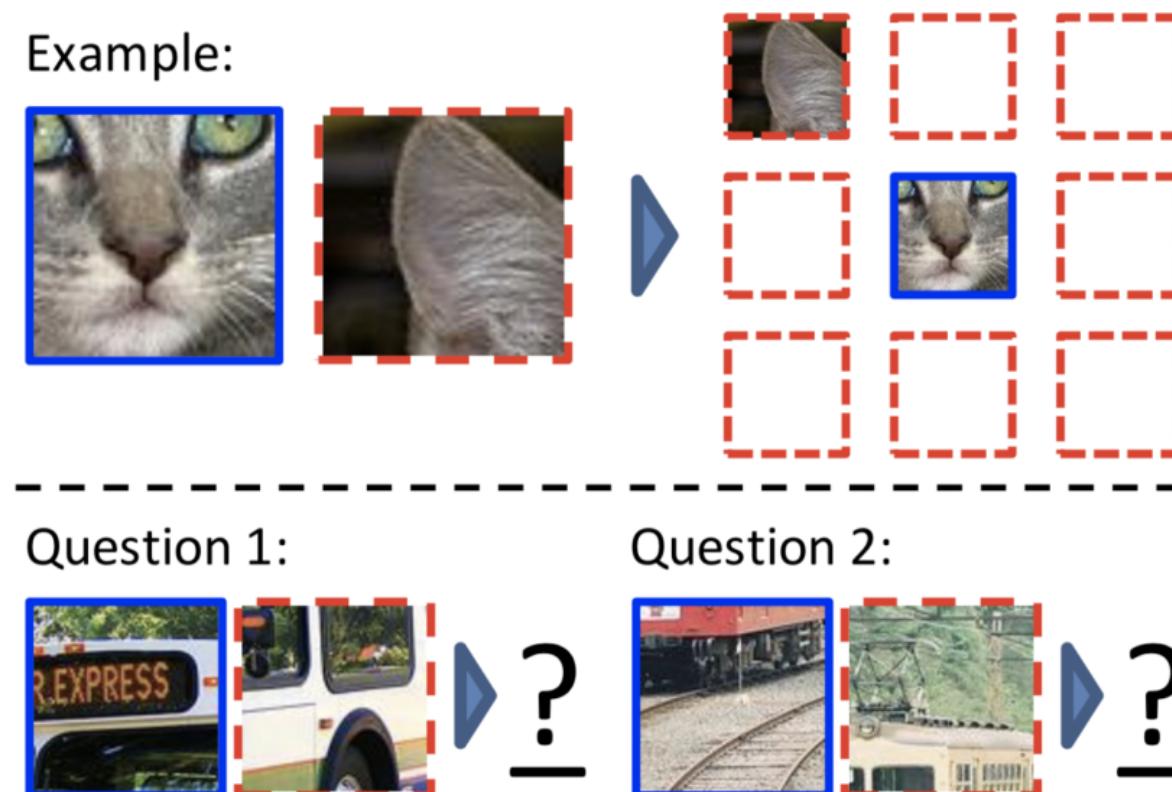


Relative position of image patches

- One can also cut two patches of an image and ask the CNN to predict their relative position on a grid.
- The task is easier when the CNN “understands” the content of the patches, i.e. has learned good features.
- Note that the two patches go through the **same** CNN, but the two outputs are concatenated before the classification layers.



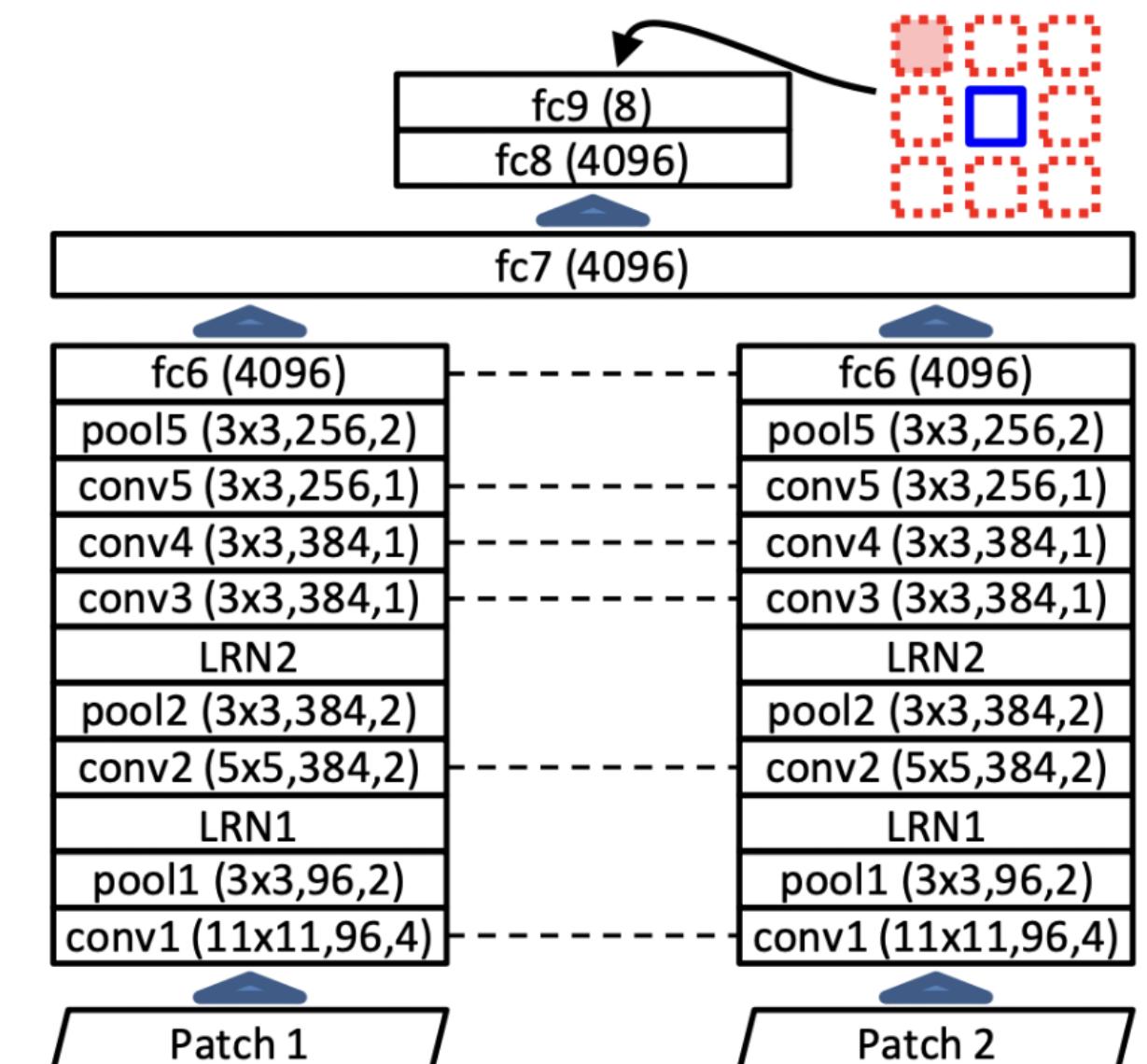
Example:



Question 1:

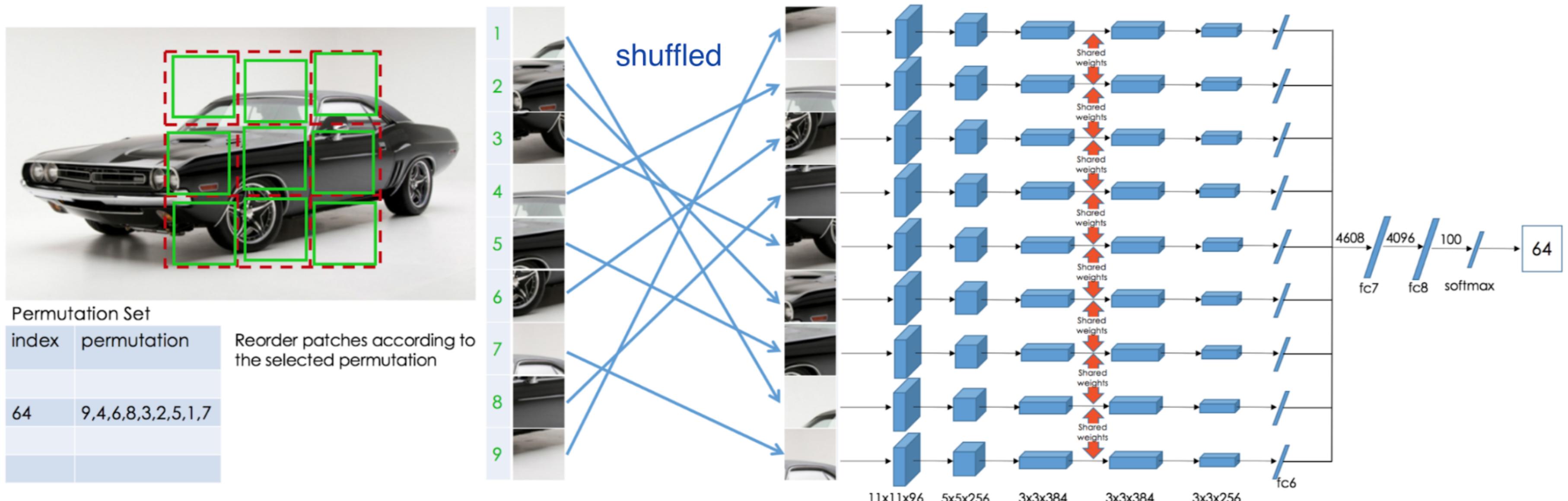


Question 2:



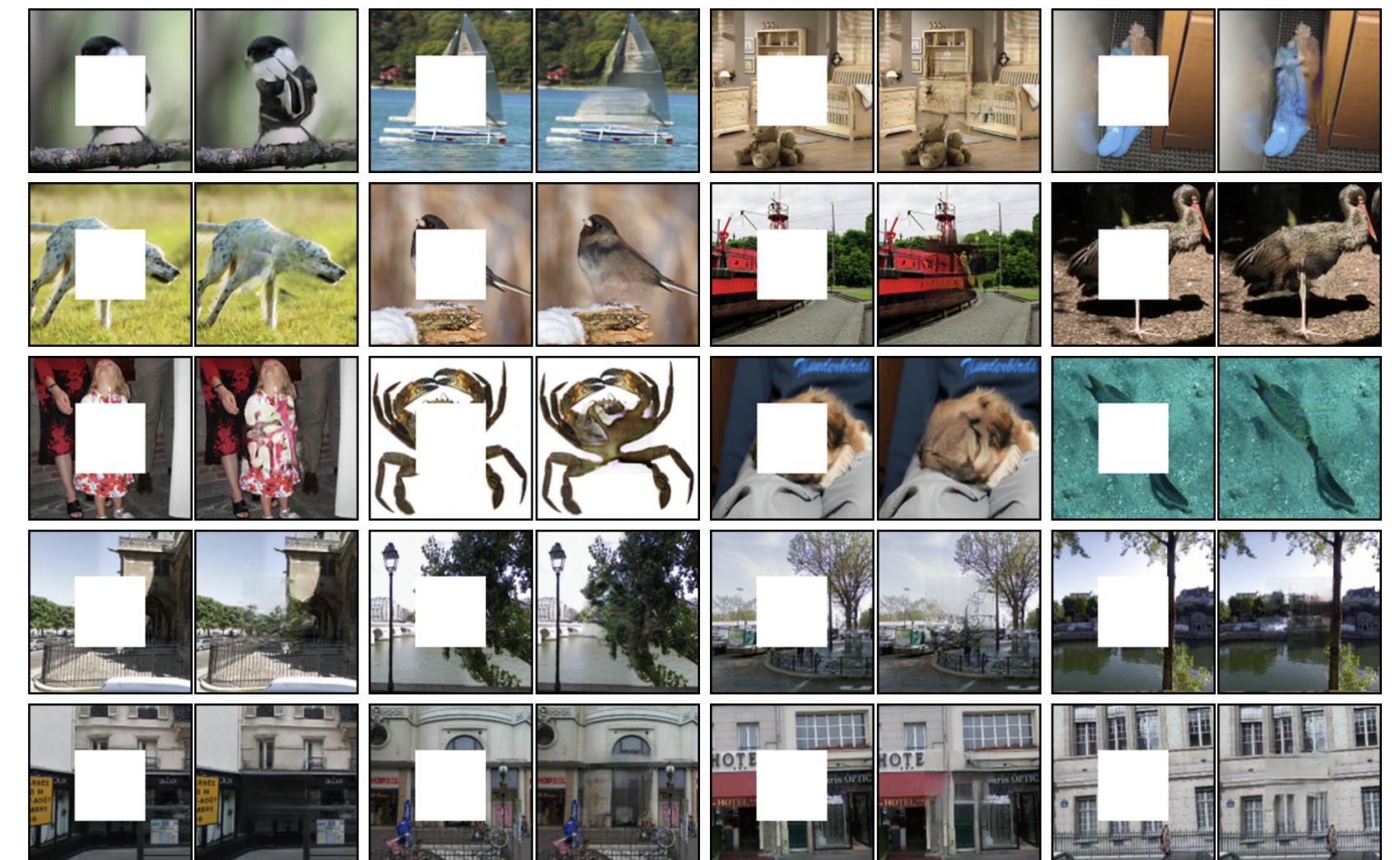
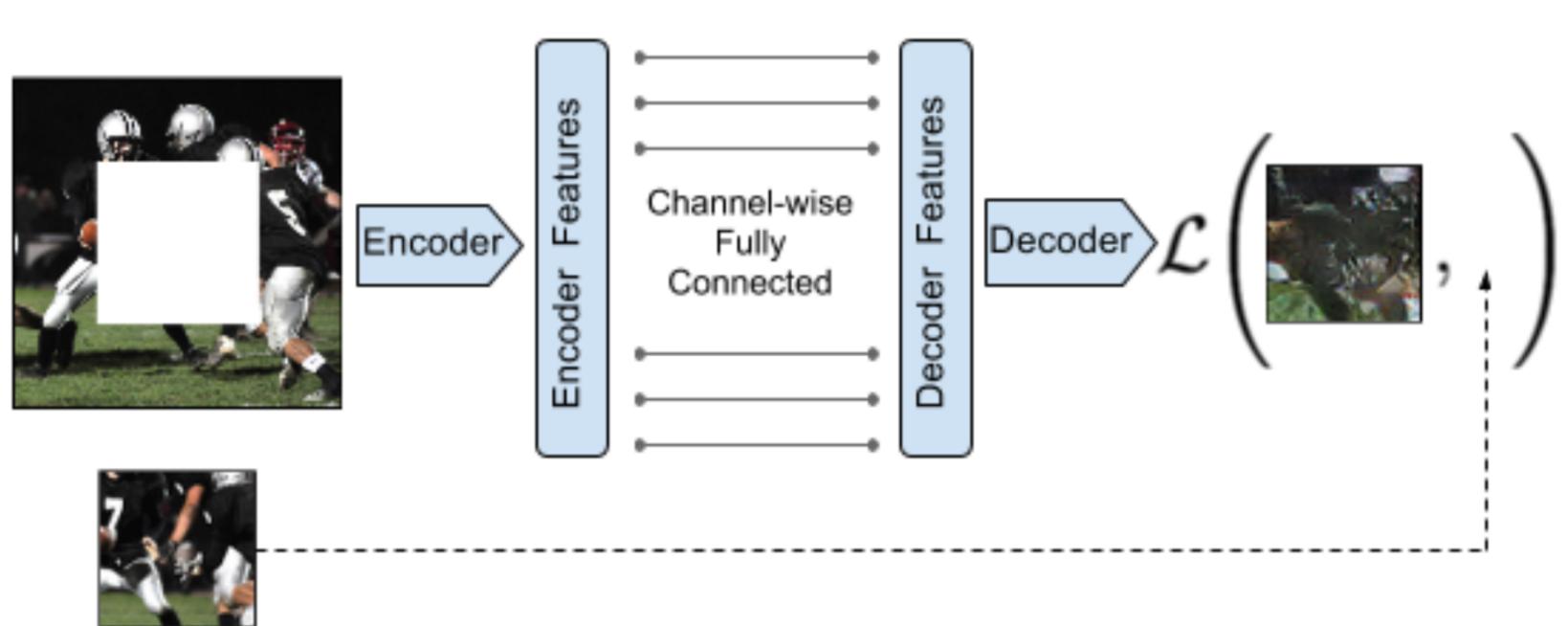
Jigsaw puzzle

- One can also shuffle patches of an image according to a specific permutation, and have the network predict which permutation was applied.



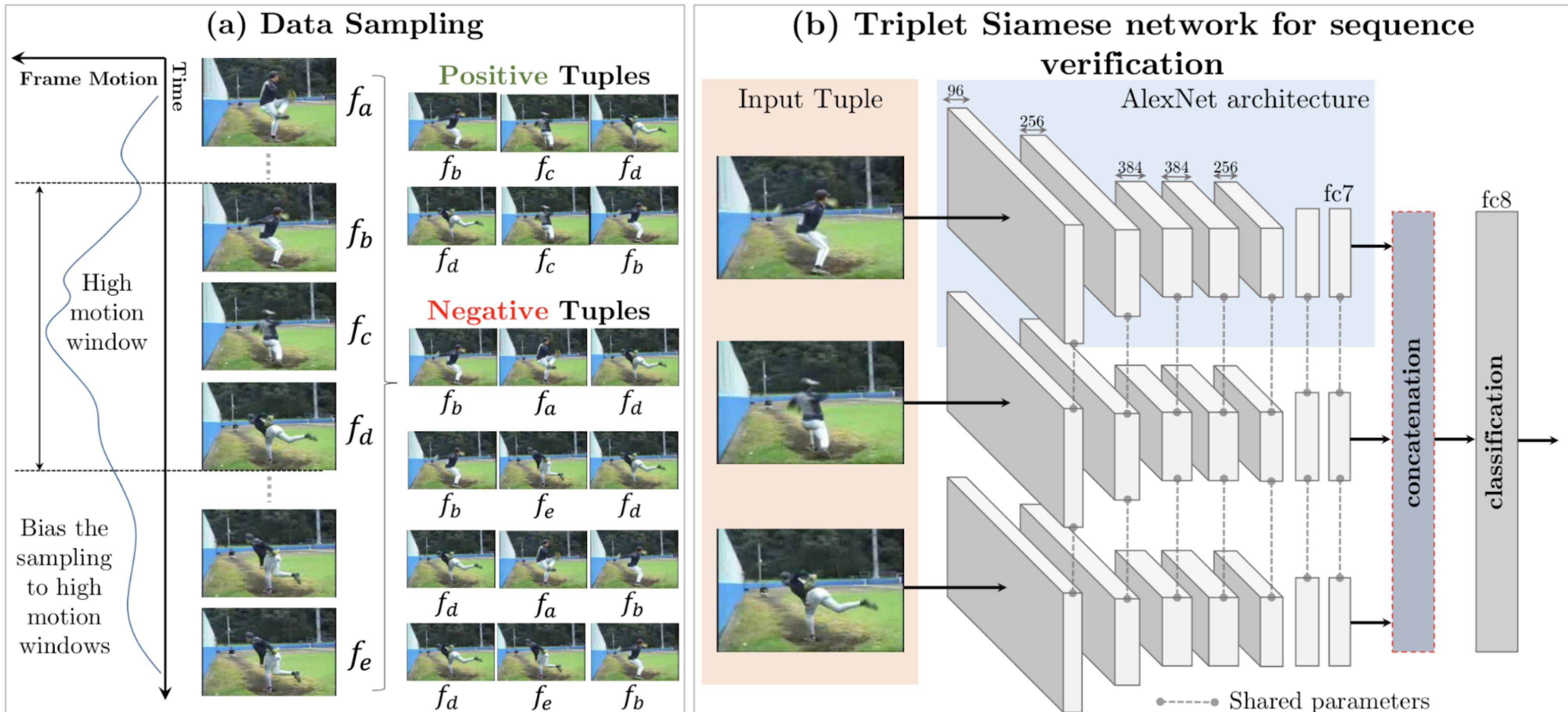
Context encoder

- As with denoising autoencoders, **context encoders** can be trained to generate the contents of an arbitrary image region based on its surroundings.
- The loss function is the sum of the reconstruction loss and an adversarial loss (as in GANs).
- Useful for in-paintings. The encoder part can be fine-tuned on classification tasks.



Frame order validation

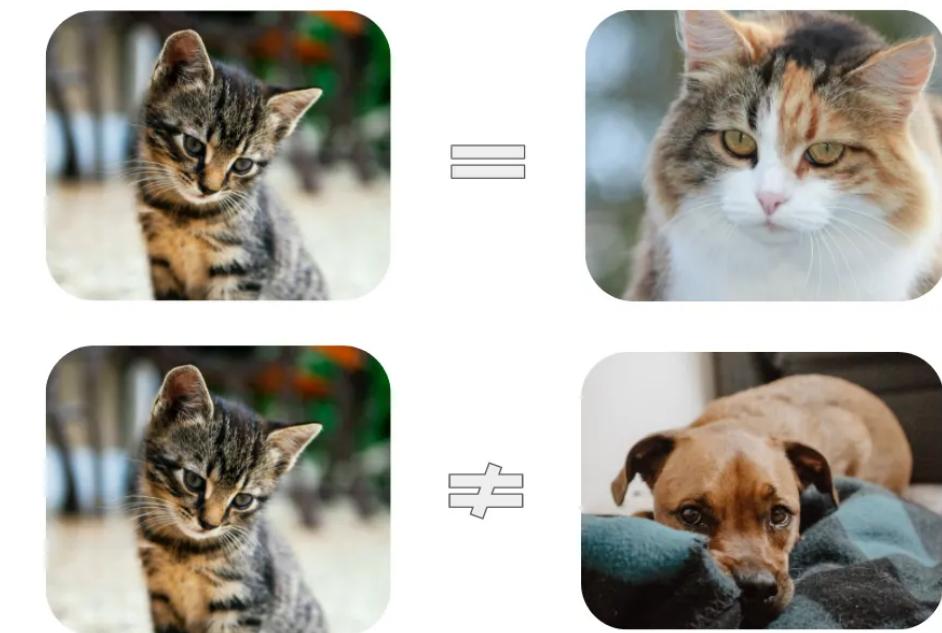
- Triplet siamese networks have to guess whether three frames are consecutive in a video sequence or not.



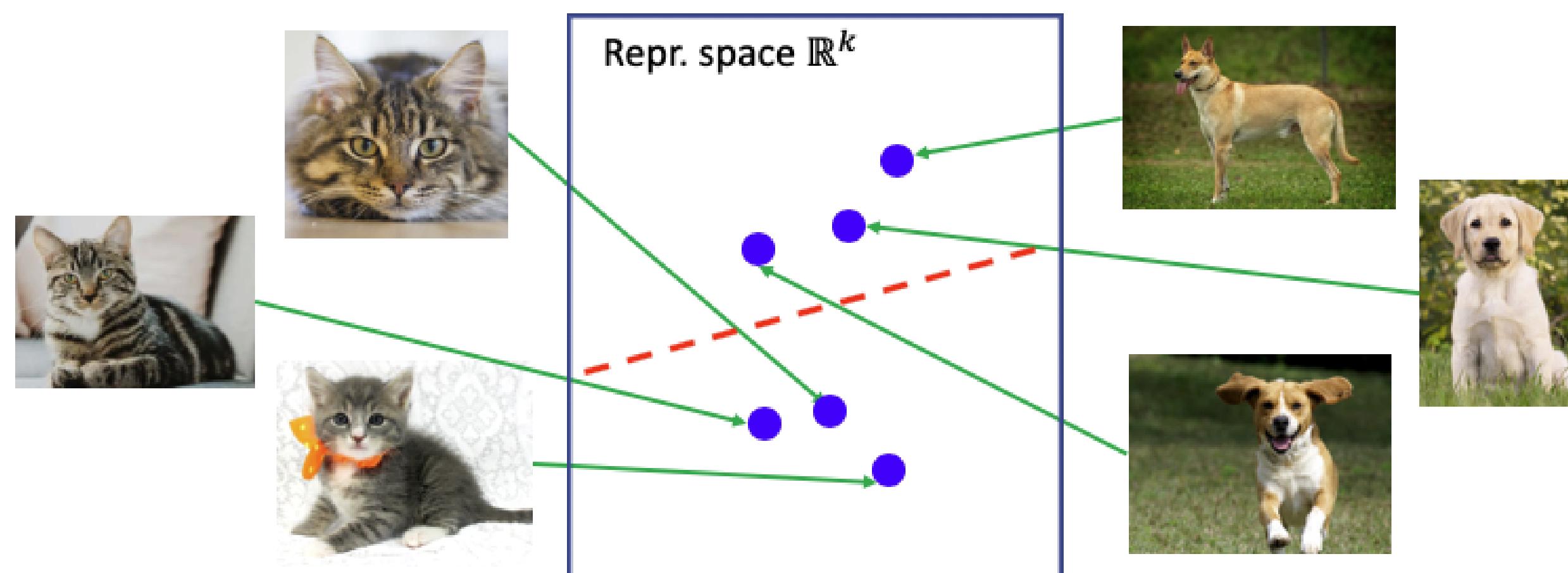
2 - Contrastive learning

Contrastive learning

- The idea of **contrastive learning** is to force a neural network to learn similar representations for similar images (e.g. cats), and different representations for different images (cats vs. dogs).
- In supervised learning, this is achieved by forcing the output layer to **linearly** separate the classes, so the last FC layer must group its representation of cats together and separate it from dogs.
- But how could we do this without the labels?



Source: <https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607>

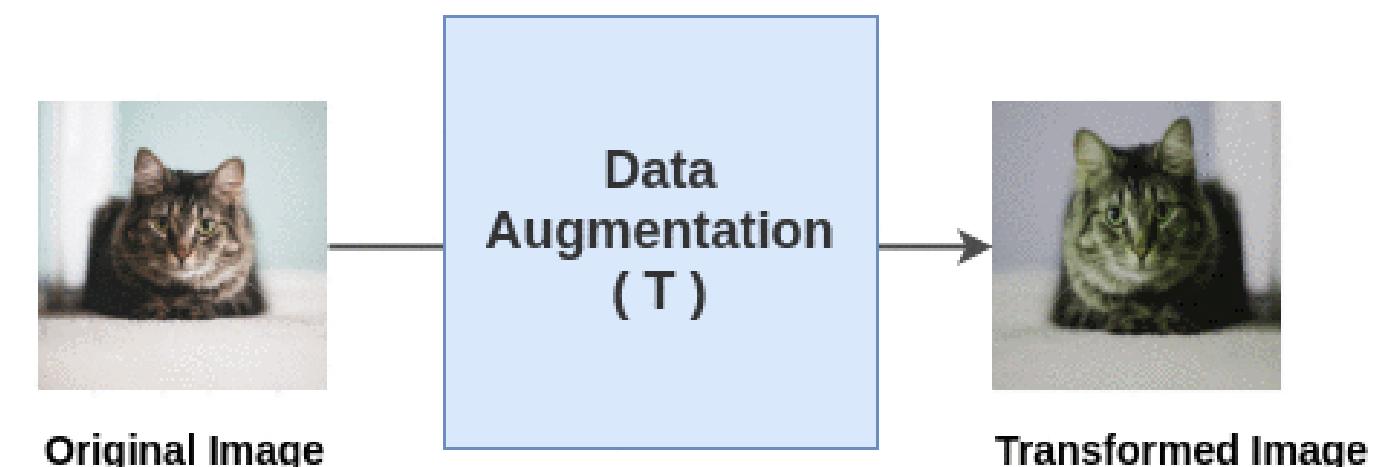


Source: <https://ai.stanford.edu/blog/understanding-contrastive-learning/>

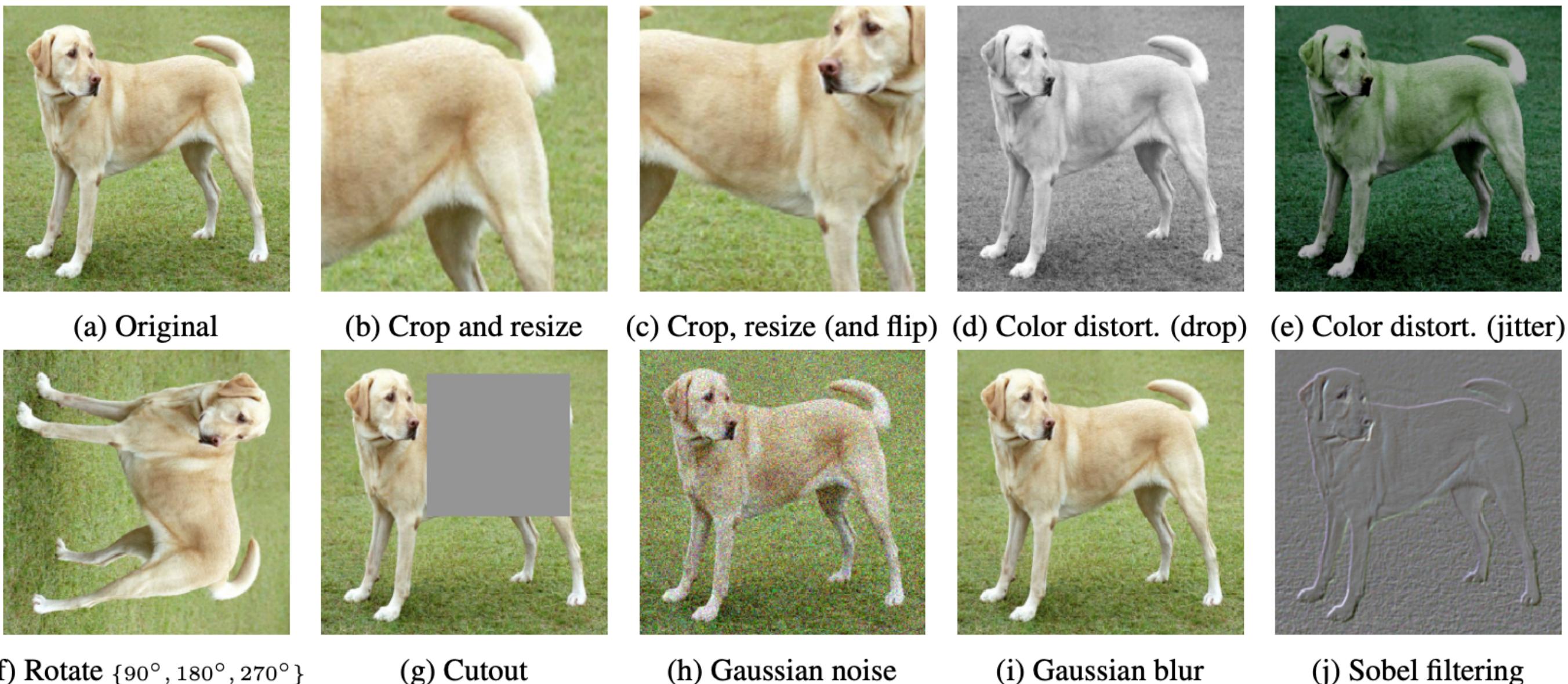
Data augmentation

- A cheap and easy to obtain similar instances of the same class without labels is to perform **data augmentation** on the same image:
 - crop, resize, flip, blur, color distortion....
- Ideally, the representation for these augmented images should be similar at the end of the neural network.

Random Transformation



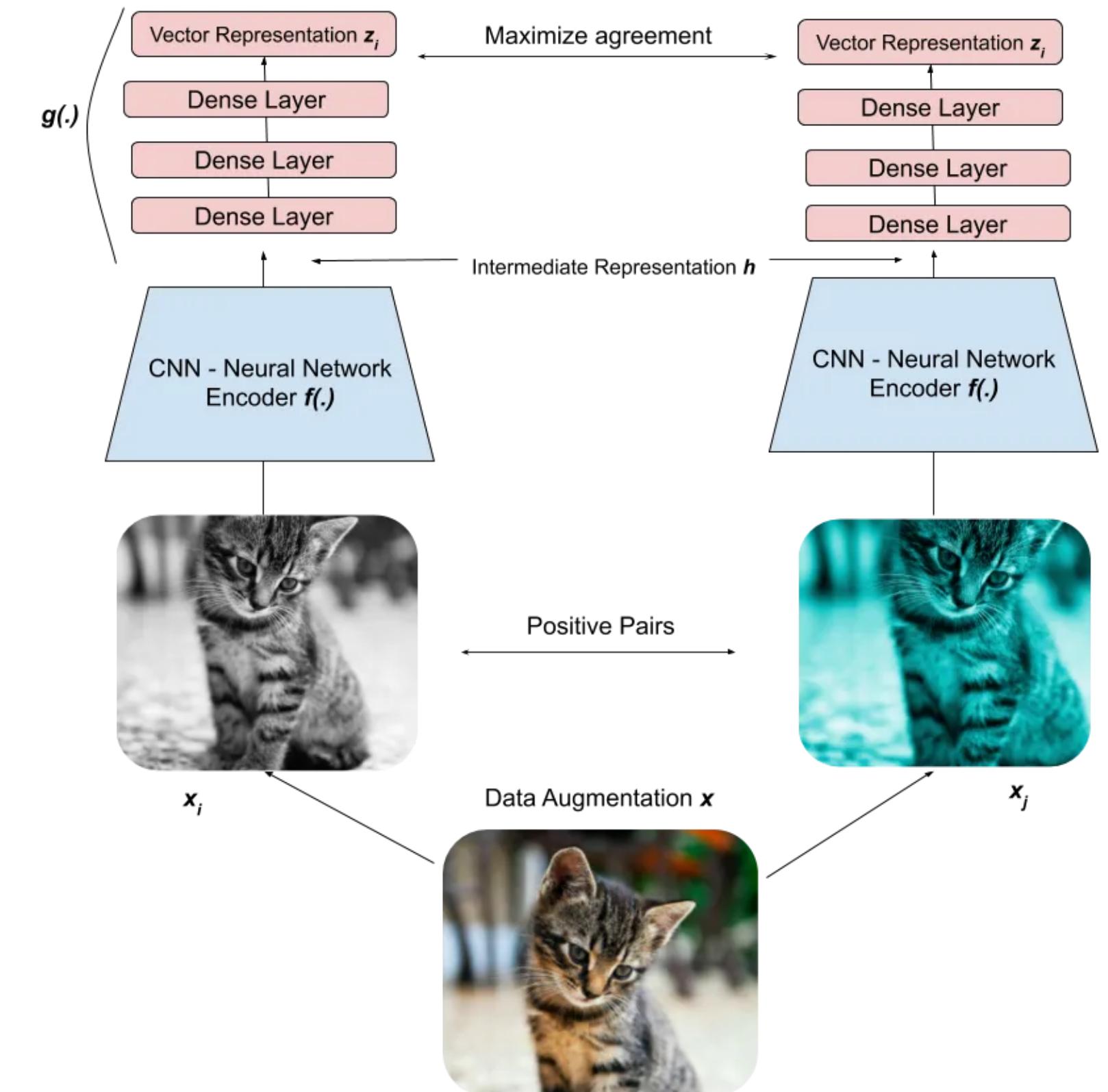
Source: <https://amitness.com/2020/03/illustrated-simclr/>



Source: Chen et al. (2020)

SimCLR

- SimCLR (**Simple framework for Contrastive Learning of visual Representations**) generates a **positive pair** of augmented images.
- Both images \mathbf{x}_i and \mathbf{x}_j go through the same CNN encoder f (e.g. a ResNet-50) to produce high-level representations \mathbf{h}_i and \mathbf{h}_j .
- The representations are passed through a FCN g to produce embeddings \mathbf{z}_i and \mathbf{z}_j .
- The goal is to **maximize the similarity** or agreement between the embeddings \mathbf{z}_i and \mathbf{z}_j , i.e. have the vectors as close as possible from each other..



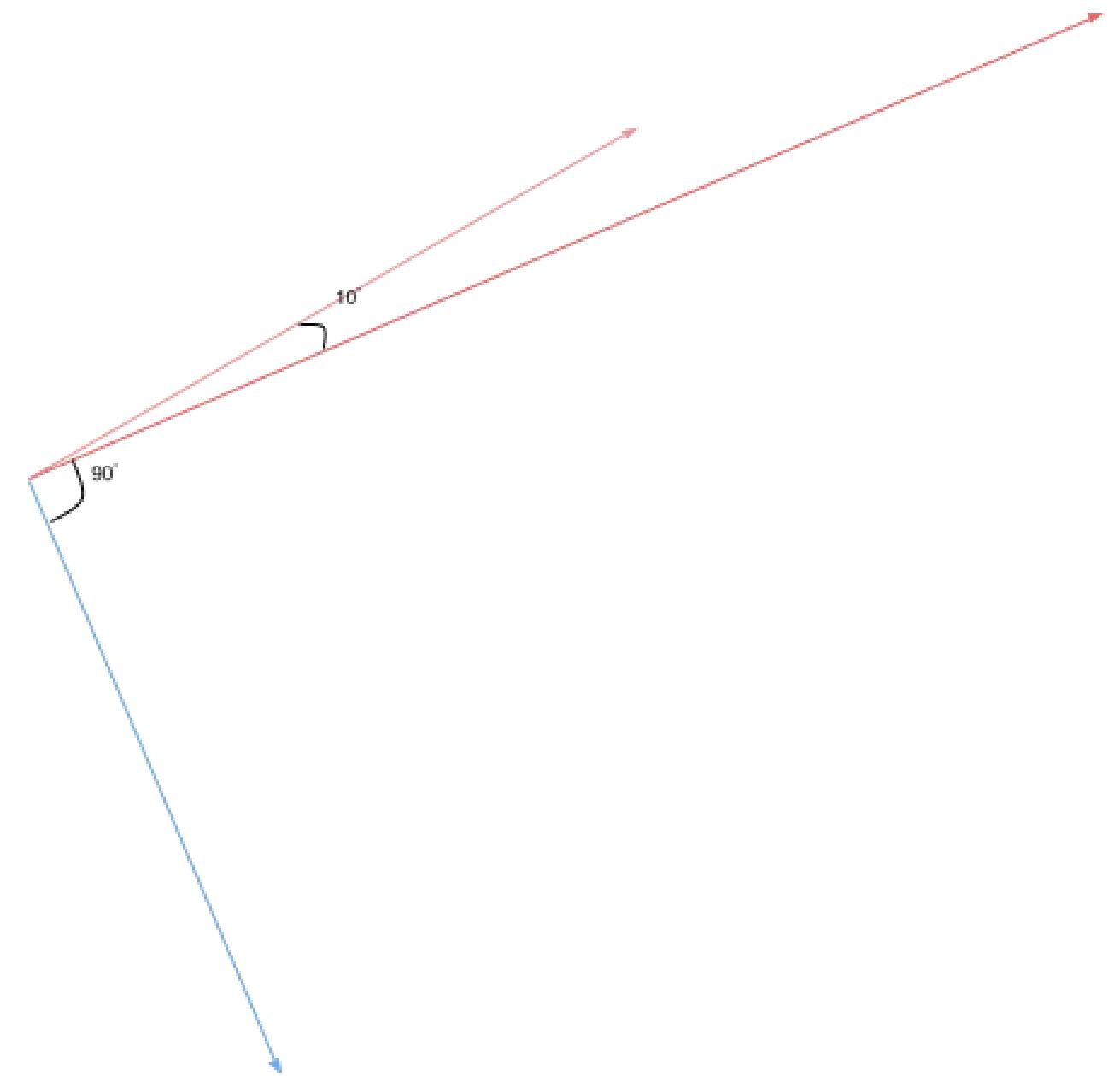
Source: <https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607>

SimCLR

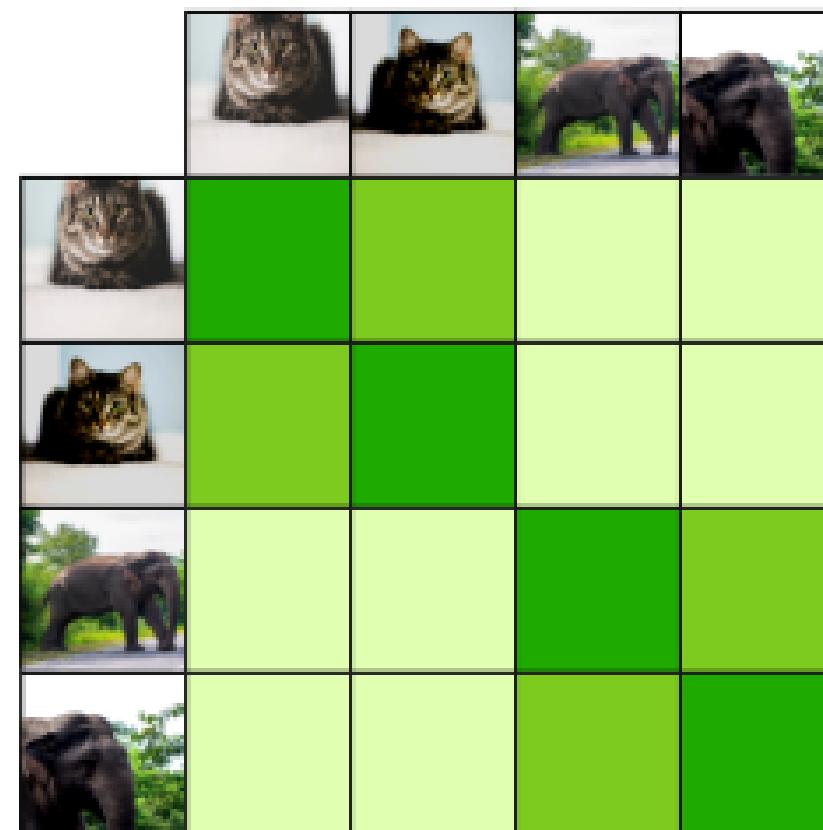
- The similarity between the embeddings \mathbf{z}_i and \mathbf{z}_j is calculated using the **cosine similarity**:

$$\cos(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^T \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|}$$

- Colinear vectors have a cosine similarity of 1 (or -1), orthogonal vector have a cosine similarity of 0.
- Note: One could use the L2-norm, but it would force the vectors to have the same norm.



Pairwise cosine similarity



Source: <https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607>

SimCLR

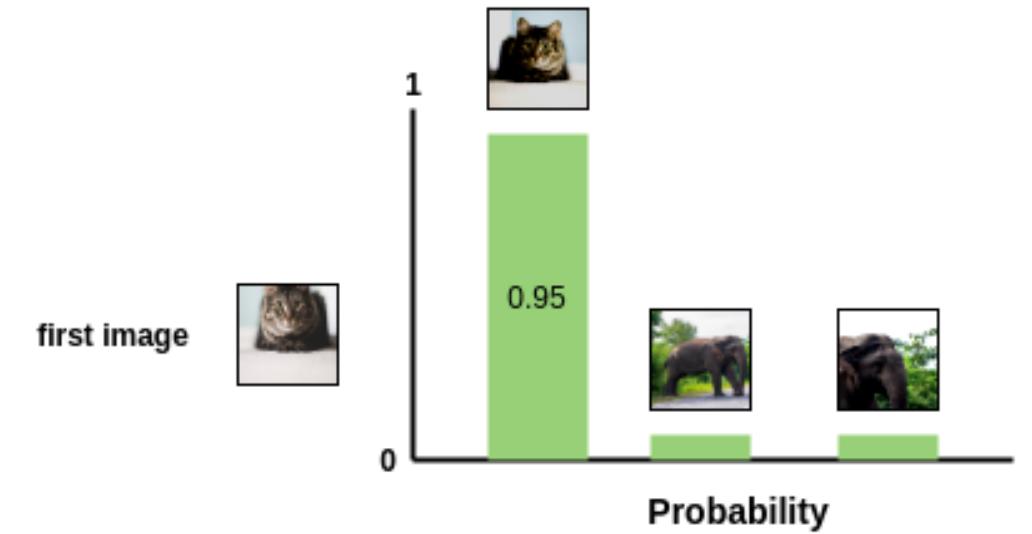
- SimCLR actually selects a minibatch of K images and generates two augmented images for each of them.
- For an image k , the goal is to:
 - **maximize** the similarity between the embeddings \mathbf{z}_{2k} and \mathbf{z}_{2k+1} of the positive pair,
 - **minimize** their similarity with the other augmented images ($(K - 1)$ negative pairs).
- There could be another instance of the same class in the minibatch, but in practice it will not matter much.
- The batch size should be quite big ($K = 8192$) to allow for many relevant negative pairs.

Source: <https://ai.googleblog.com/2020/04/advancing-self-supervised-and-semi.html>

SimCLR

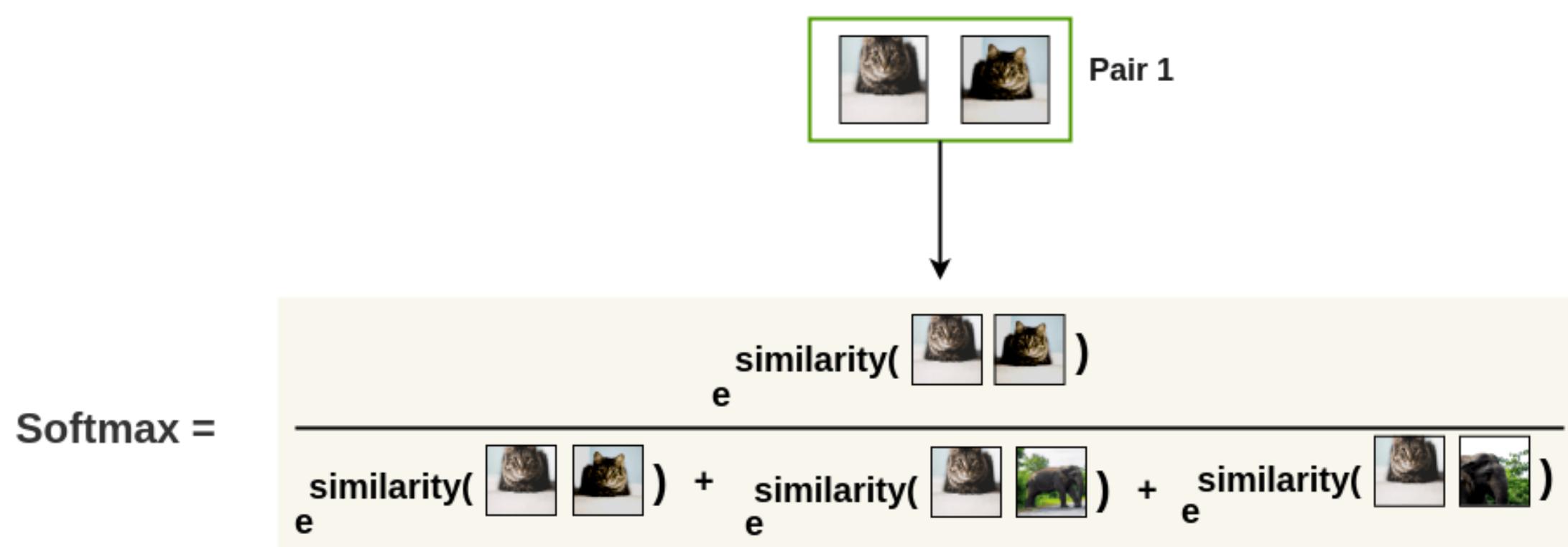
- The NT-Xent (Normalized Temperature-Scaled Cross-Entropy Loss) loss function allows to achieve this. It is a variant of the Noise Contrastive Estimator (NCE) loss.
- Let's first transform the cosine similarity between two images i and j into a probability using a softmax:

$$s(i, j) = \frac{\exp \frac{\cos(\mathbf{z}_i, \mathbf{z}_j)}{\tau}}{\sum_{k \neq i} \exp \frac{\cos(\mathbf{z}_i, \mathbf{z}_k)}{\tau}}$$



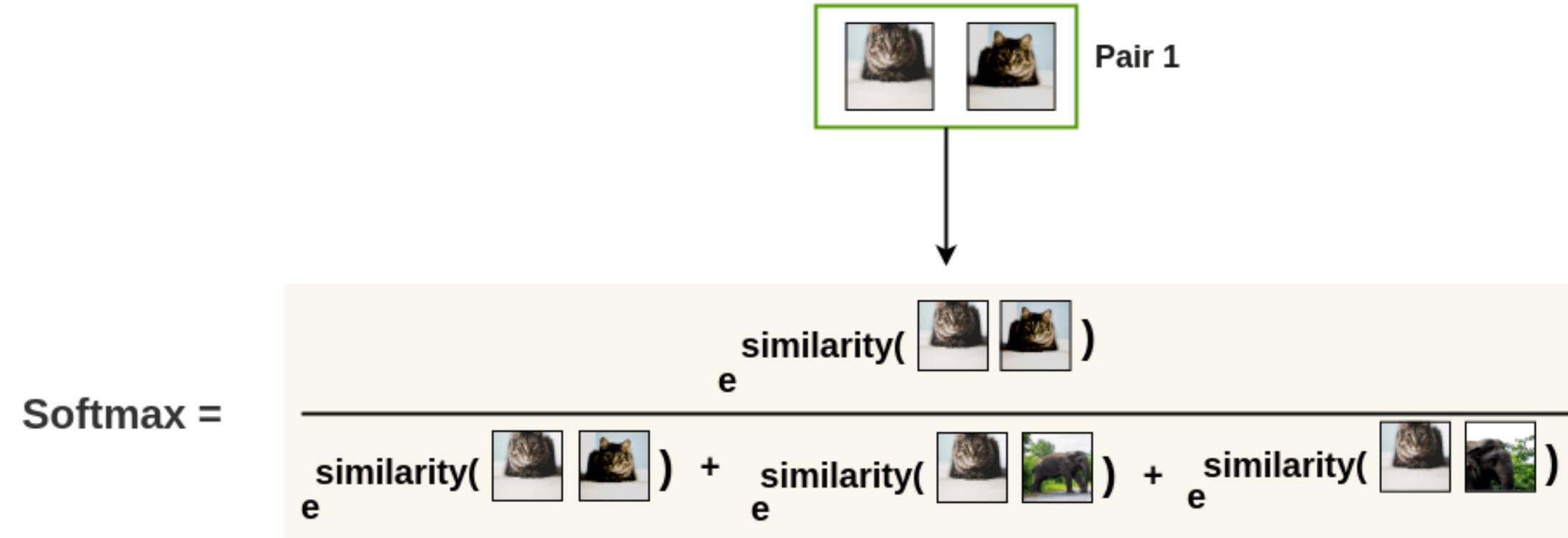
Source:
<https://amitness.com/2020/03/illustrated-simclr/>

- For a positive pair, this softmax represents the probability that the second augmented cat is closer to the first one, compared to the other negative images in the minibatch:



SimCLR

- By maximizing this probability for a positive pair, we not only maximize the similarity between them, but we also minimize the similarity with the negative pairs, as they appear at the denominator!



- In practice, we will minimize the **negative log-likelihood**:

$$l(i, j) = -\log s(i, j)$$

The diagram shows a formula for the negative log-likelihood loss function:

$$l(\text{Pair 1}) = -\log \left(\frac{\text{e}^{\text{similarity}(\text{Pair 1})}}{\text{e}^{\text{similarity}(\text{Pair 1})} + \text{e}^{\text{similarity}(\text{cat, elephant})} + \text{e}^{\text{similarity}(\text{cat, bear})}} \right)$$

Source: <https://amitness.com/2020/03/illustrated-simclr/>

SimCLR

- Note that the loss function is not symmetric, as the denominator changes:

$$l(\begin{array}{c} \text{cat} \\ \text{dog} \end{array}, \begin{array}{c} \text{cat} \\ \text{dog} \end{array}) = -\log \left(\frac{e^{\text{similarity}(\begin{array}{c} \text{cat} \\ \text{dog} \end{array}, \begin{array}{c} \text{cat} \\ \text{dog} \end{array})}}{e^{\text{similarity}(\begin{array}{c} \text{cat} \\ \text{dog} \end{array}, \begin{array}{c} \text{cat} \\ \text{dog} \end{array})} + e^{\text{similarity}(\begin{array}{c} \text{cat} \\ \text{dog} \end{array}, \begin{array}{c} \text{elephant} \\ \text{dog} \end{array})} + e^{\text{similarity}(\begin{array}{c} \text{cat} \\ \text{dog} \end{array}, \begin{array}{c} \text{elephant} \\ \text{dog} \end{array})}} \right)$$

Interchanged

$$l(\begin{array}{c} \text{cat} \\ \text{dog} \end{array}, \begin{array}{c} \text{dog} \\ \text{cat} \end{array}) = -\log \left(\frac{e^{\text{similarity}(\begin{array}{c} \text{cat} \\ \text{dog} \end{array}, \begin{array}{c} \text{dog} \\ \text{cat} \end{array})}}{e^{\text{similarity}(\begin{array}{c} \text{cat} \\ \text{dog} \end{array}, \begin{array}{c} \text{dog} \\ \text{cat} \end{array})} + e^{\text{similarity}(\begin{array}{c} \text{cat} \\ \text{dog} \end{array}, \begin{array}{c} \text{elephant} \\ \text{dog} \end{array})} + e^{\text{similarity}(\begin{array}{c} \text{cat} \\ \text{dog} \end{array}, \begin{array}{c} \text{elephant} \\ \text{dog} \end{array})}} \right)$$

Source: <https://amitness.com/2020/03/illustrated-simclr/>

- For a positive pair $(2k, 2k + 1)$, we will then average the two losses:

$$l(k) = \frac{-\log s(2k, 2k + 1) - \log s(2k + 1, 2k)}{2}$$

SimCLR

- Finally, we sum over all positive pairs in the minibatch to obtain the **NT-Xent** (Normalized Temperature-Scaled Cross-Entropy Loss) loss:

$$\mathcal{L}(\theta) = -\frac{1}{2K} \sum_{k=1}^K \log s(2k, 2k+1) + \log s(2k+1, 2k)$$

$$L = \frac{[I(\text{cat}, \text{cat}) + I(\text{cat}, \text{cat})] + [I(\text{elephant}, \text{elephant}) + I(\text{elephant}, \text{elephant})]}{2 * 2}$$

Pair 1 Loss (k=1) Pair 2 Loss (k=2)

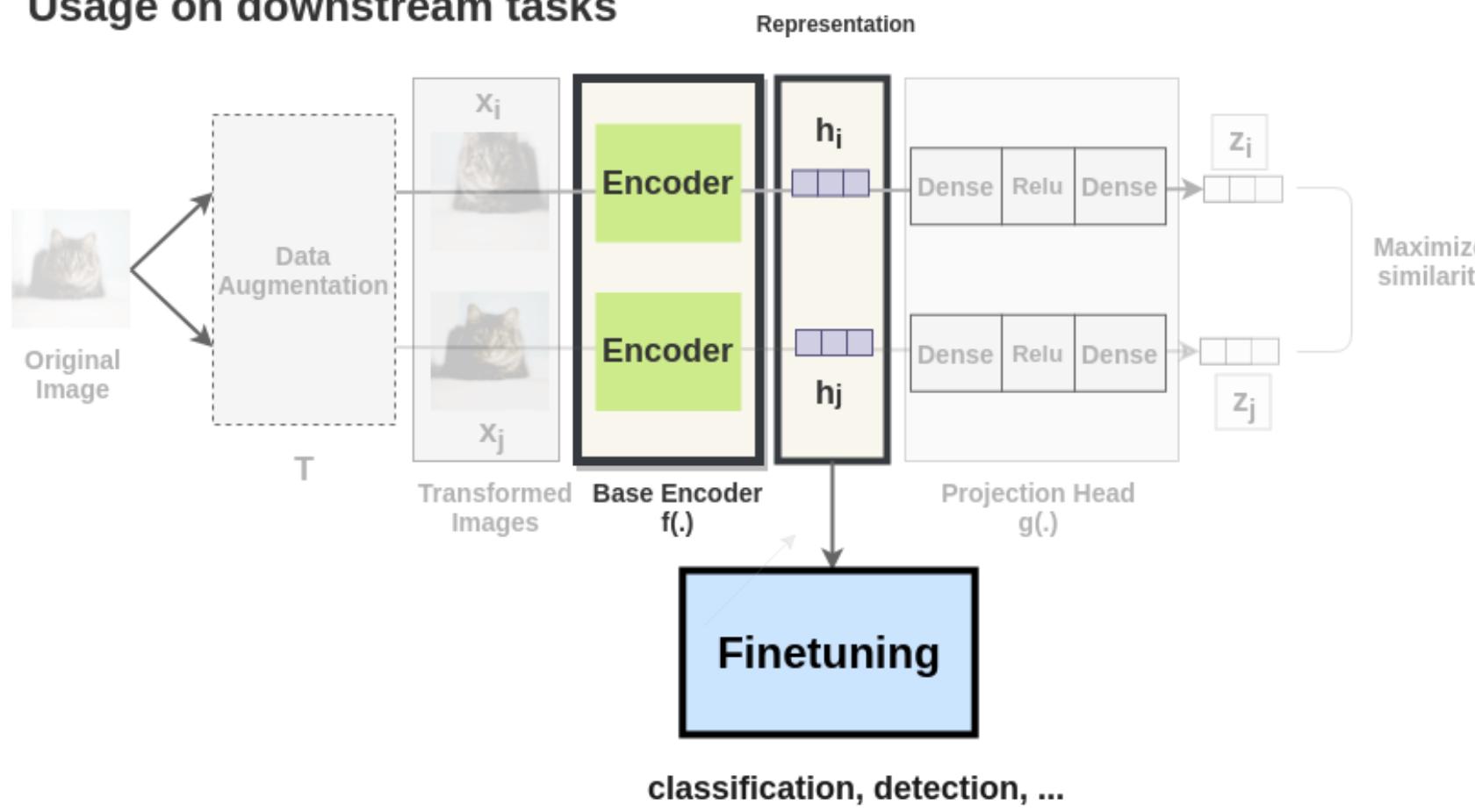


- The loss is defined only over positive pairs in the minibatch, but the negative pairs influence it through the softmax.
- The temperature plays an important role and should be adapted to the batch size and the number of epochs.

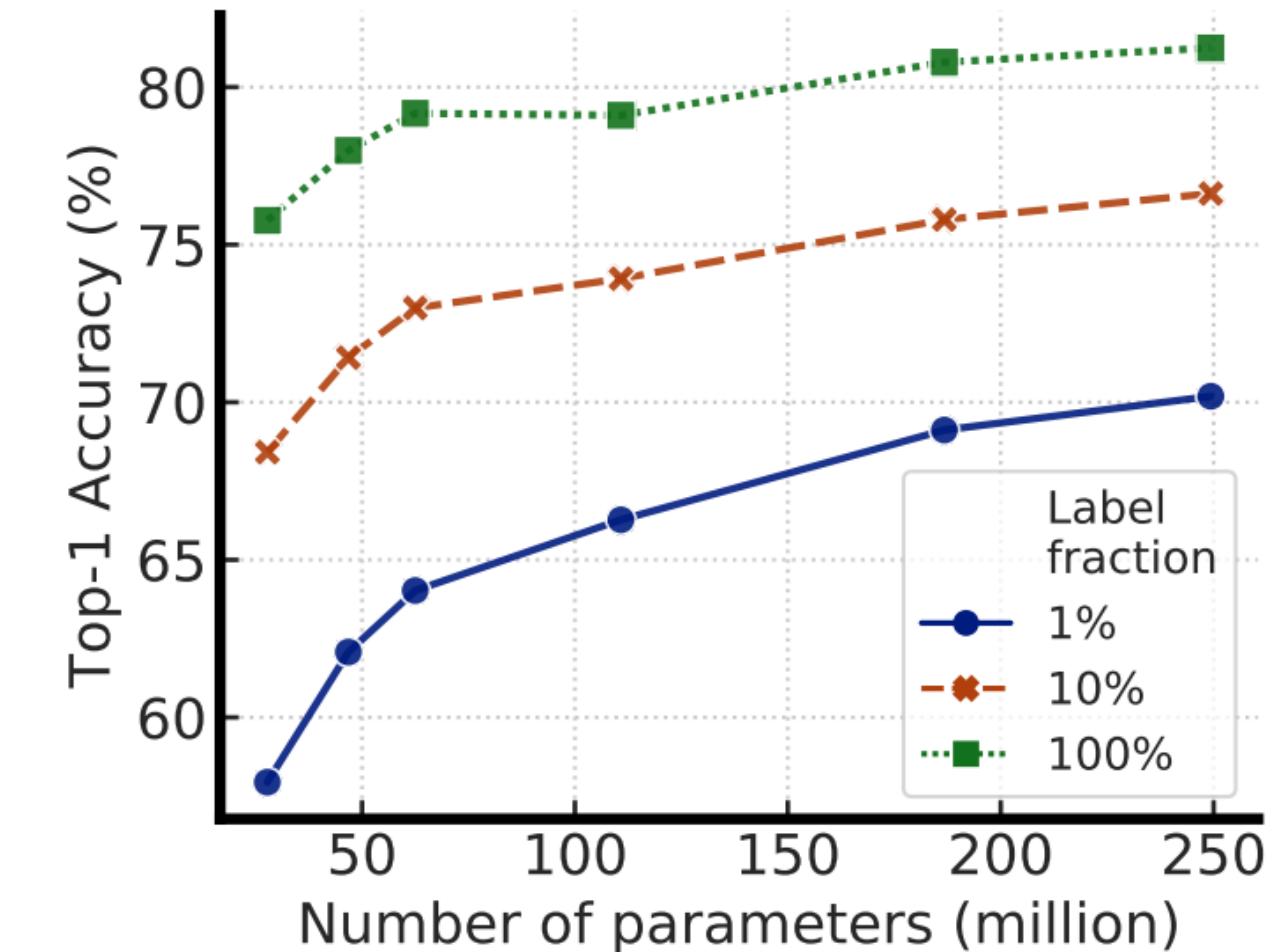
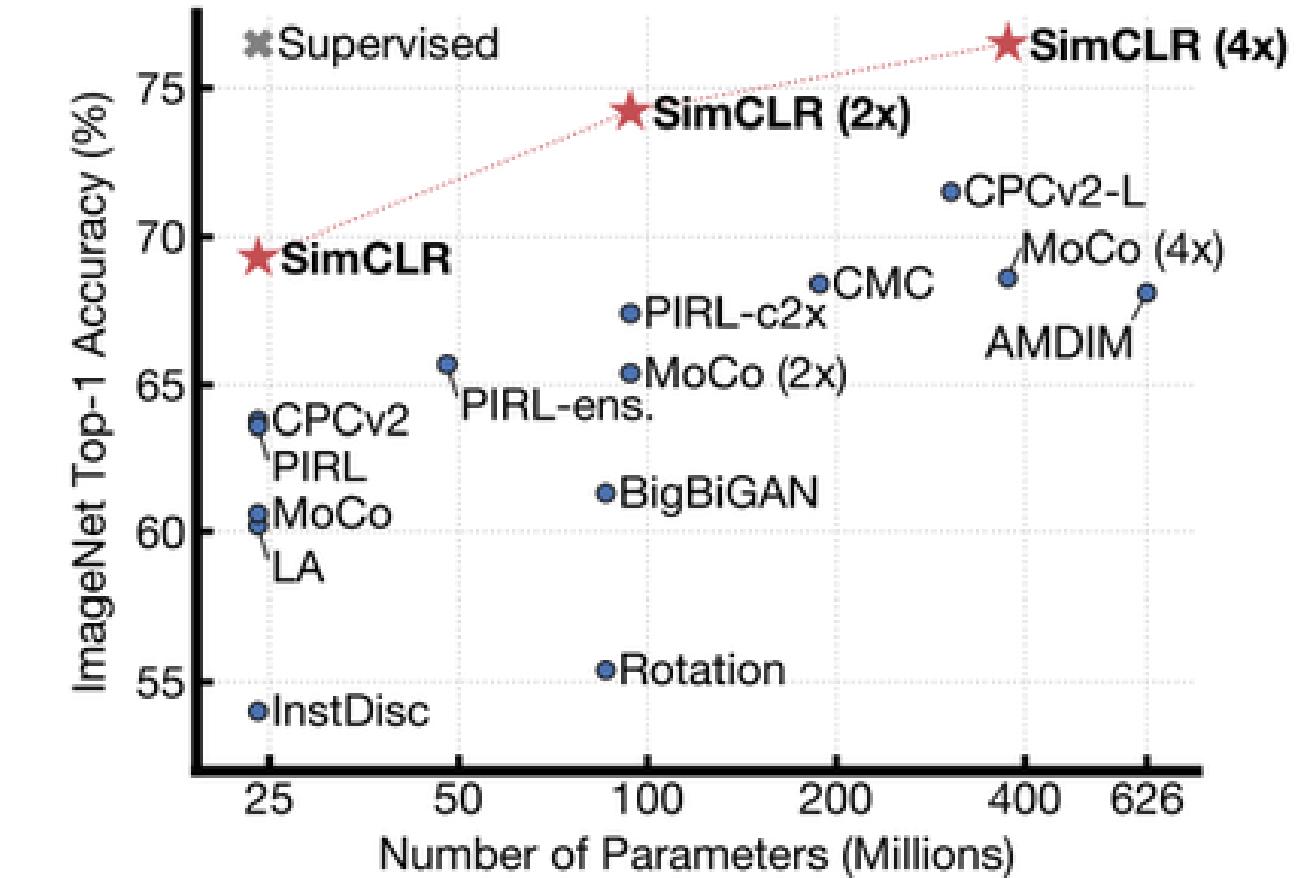
SimCLR

- After performing contrastive learning on the training set of ImageNet (or bigger), the Resnet-50 encoder can be used to:
 - linearly predict the labels using logistic regression.
 - fine-tune on 1% of the training data.
- A simple logistic regression on the learned representations is already on-par with fully supervised models.

Usage on downstream tasks



Source: <https://amitness.com/2020/03/illustrated-simclr>



(c) Semi-supervised (y-axis zoomed)

Additional resources

<https://amitness.com/2020/03/illustrated-simclr/>

<https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607>

<https://lilianweng.github.io/posts/2019-11-10-self-supervised/>

<https://lilianweng.github.io/posts/2021-05-31-contrastive>

https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial17/SimCLR.html

https://docs.google.com/presentation/d/1ccddJFD_j3p3h0TCqSV9ajSi2y1y0fh0-IJoK29ircs

<https://sthalles.github.io/simple-self-supervised-learning/>

References

- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations.
doi:10.48550/arXiv.2002.05709.
- Doersch, C., Gupta, A., and Efros, A. A. (2016). Unsupervised Visual Representation Learning by Context Prediction. doi:10.48550/arXiv.1505.05192.
- Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised Representation Learning by Predicting Image Rotations. doi:10.48550/arXiv.1803.07728.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. <http://arxiv.org/abs/1301.3781>.
- Misra, I., Zitnick, C. L., and Hebert, M. (2016). Shuffle and Learn: Unsupervised Learning using Temporal Order Verification.
doi:10.48550/arXiv.1603.08561.
- Noroozi, M., and Favaro, P. (2017). Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. doi:10.48550/arXiv.1603.09246.
- Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context Encoders: Feature Learning by Inpainting.
doi:10.48550/arXiv.1604.07379.
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 38.