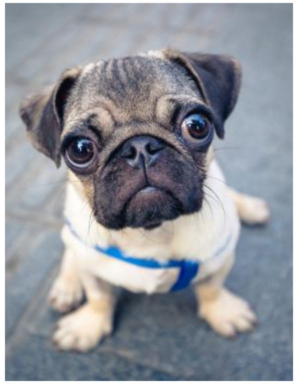
SELF-SUPERVISED LEARNING

MACHINE LEARNING FOR IMAGING - 09.02.2024

French Bulldog



Pug

Supervised classifiers are very **data hungry**

In some domains, e.g. healthcare, **labels are very expensive to acquire**, but we have a lot of unlabelled images available…

→ Could we also learn to understand images without any labels?

STANDARD SUPERVISED CLASSIFICATION PIPELINE

Encoder

h

Classification head

P(Cat) = 0.9

Trained with cross-entropy loss with the labels

Outputs a vector h summarising the information in the image useful for the task

Classifies the image from the learned representation h

STANDARD SUPERVISED CLASSIFICATION PIPELINE

Can we also learn useful image representation *h* without labels ?

Encoder $\xrightarrow{h}$ Classification head

P(Cat) = 0.9

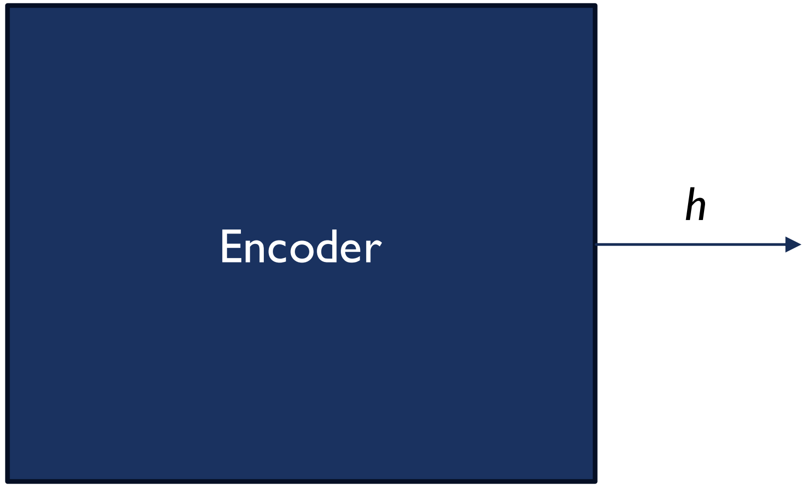
Trained with cross-entropy loss with the labels

Outputs a vector *h* summarising the information in the image useful for the task

Classifies the image from the learned representation *h*

SELF-SUPERVISED LEARNING: USING THE IMAGE ITSELF AS SUPERVISION SIGNAL

Can we also learn useful image representation *h* without labels ?



Outputs a vector *h* summarising the information in the image useful for the task

SELF-SUPERVISED LEARNING: USING THE IMAGE ITSELF AS SUPERVISION SIGNAL

Encoder

Outputs a vector *h* summarising the information in the image useful for the task

We want to learn a representation h, that summarises the main information in the image without needing any external labels.

→ **We need to create "proxy" tasks that only require the images to create the supervision signal**

SELF-SUPERVISED LEARNING: USING THE IMAGE ITSELF AS SUPERVISION SIGNAL

How can we create synthetic tasks that will get the network to learn useful representation ?

Any ideas?

SELF-SUPERVISED LEARNING: USING THE IMAGE ITSELF AS SUPERVISION SIGNAL

- How can we create synthetic tasks that will get the network to learn useful representation ?

  → There are many different paradigms, each defining different approaches to self-supervised learning

- In this lecture we will look at three different paradigms:

  Contrastive-based learning: teaching the network to recognise meaningful pairs of images

  Generative-based learning: teaching the network to reconstruct an image from a corrupted version.

  Joint-Embedding Prediction: a mix of both

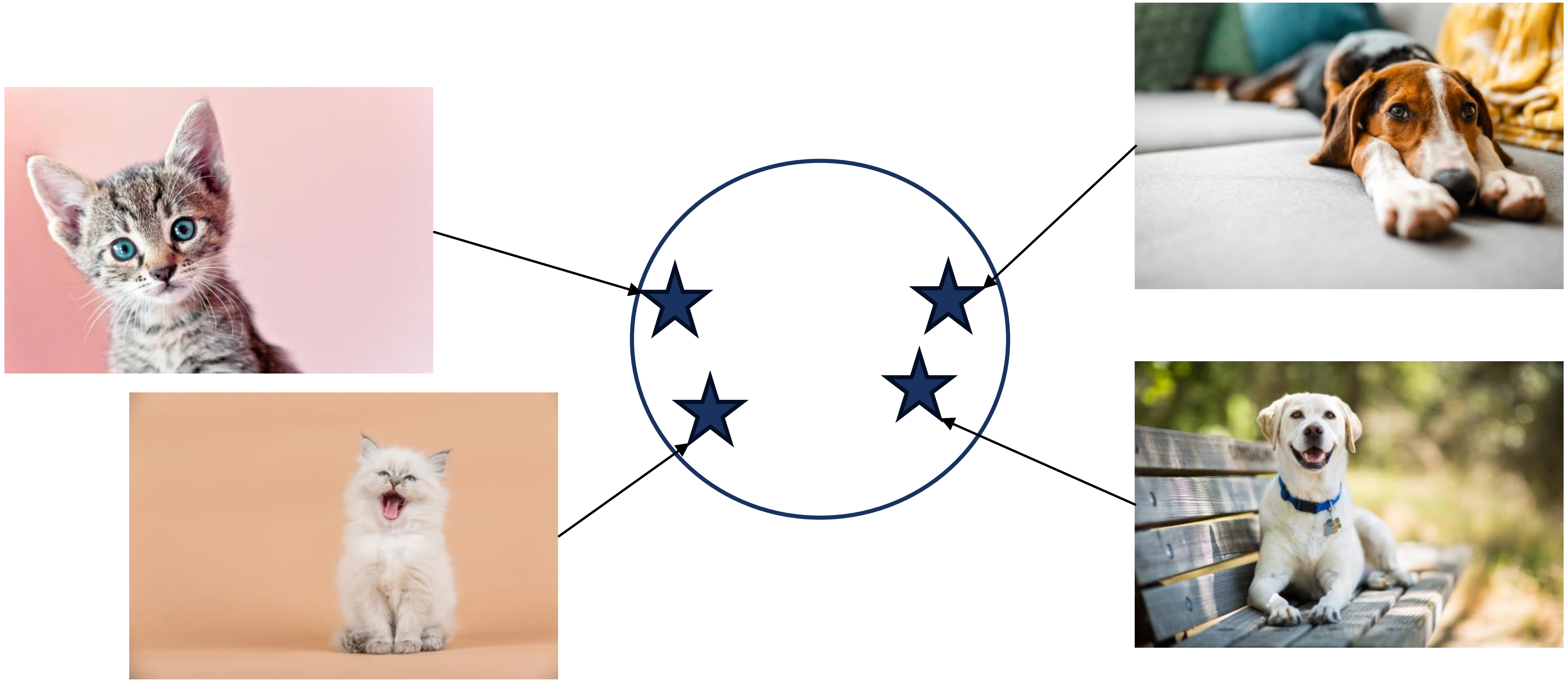
CONTRASTIVE LEARNING

CONTRASTIVE LEARNING – THE MAIN IDEA

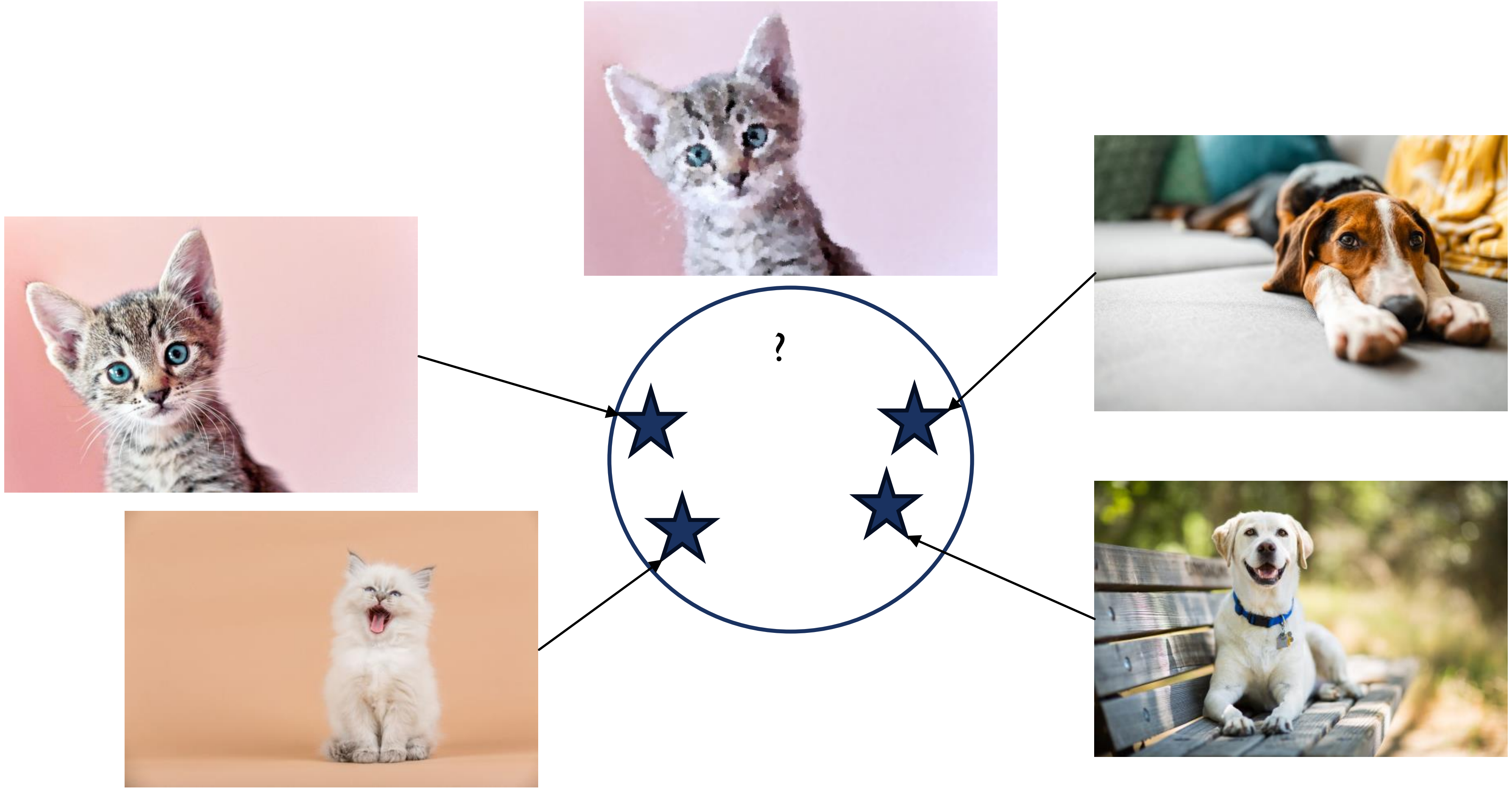
→ A meaningful representation should put similar images closer to each other

CONTRASTIVE LEARNING – THE MAIN IDEA

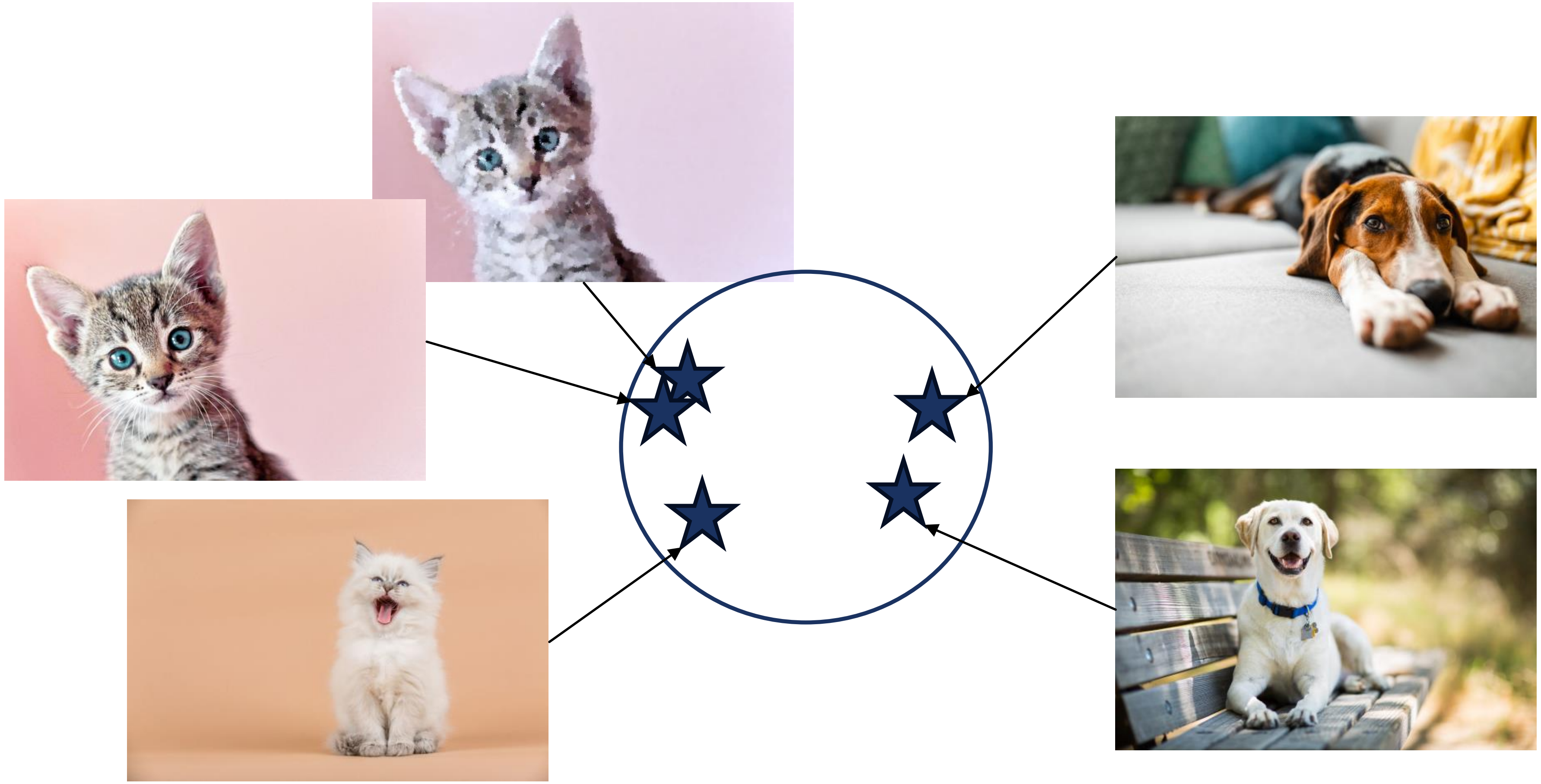
→ Without labels I don't know which images are dogs or cats so I can not use this information as supervision signal



CONTRASTIVE LEARNING – THE MAIN IDEA

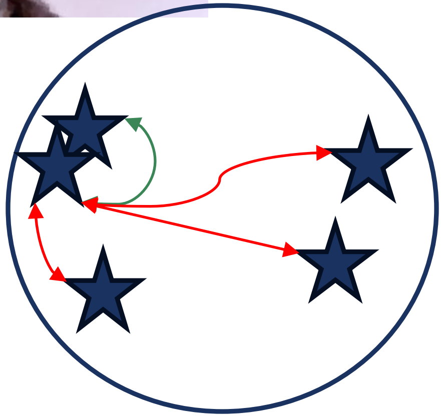


CONTRASTIVE LEARNING – THE MAIN IDEA



→ If I artificially create a **perturbed version of the same image**, I know **its representation should be closest to the original image** compared to all other images.

CONTRASTIVE LEARNING – THE MAIN IDEA



→This is the main principle of "contrastive learning"

→Create multiple versions of the same image and teach the network to recognise the correct pair

SIMCLR

A Simple Framework for Contrastive Learning of Visual Representations

Ting Chen [1] **Simon Kornblith** [1] **Mohammad Norouzi** [1] **Geoffrey Hinton** [1]

Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *International conference on machine learning*. PMLR, 2020.
http://proceedings.mlr.press/v119/chen20j/chen20j.pdf

SIMCLR - MAIN COMPONENTS

Encoder
f(.)

h

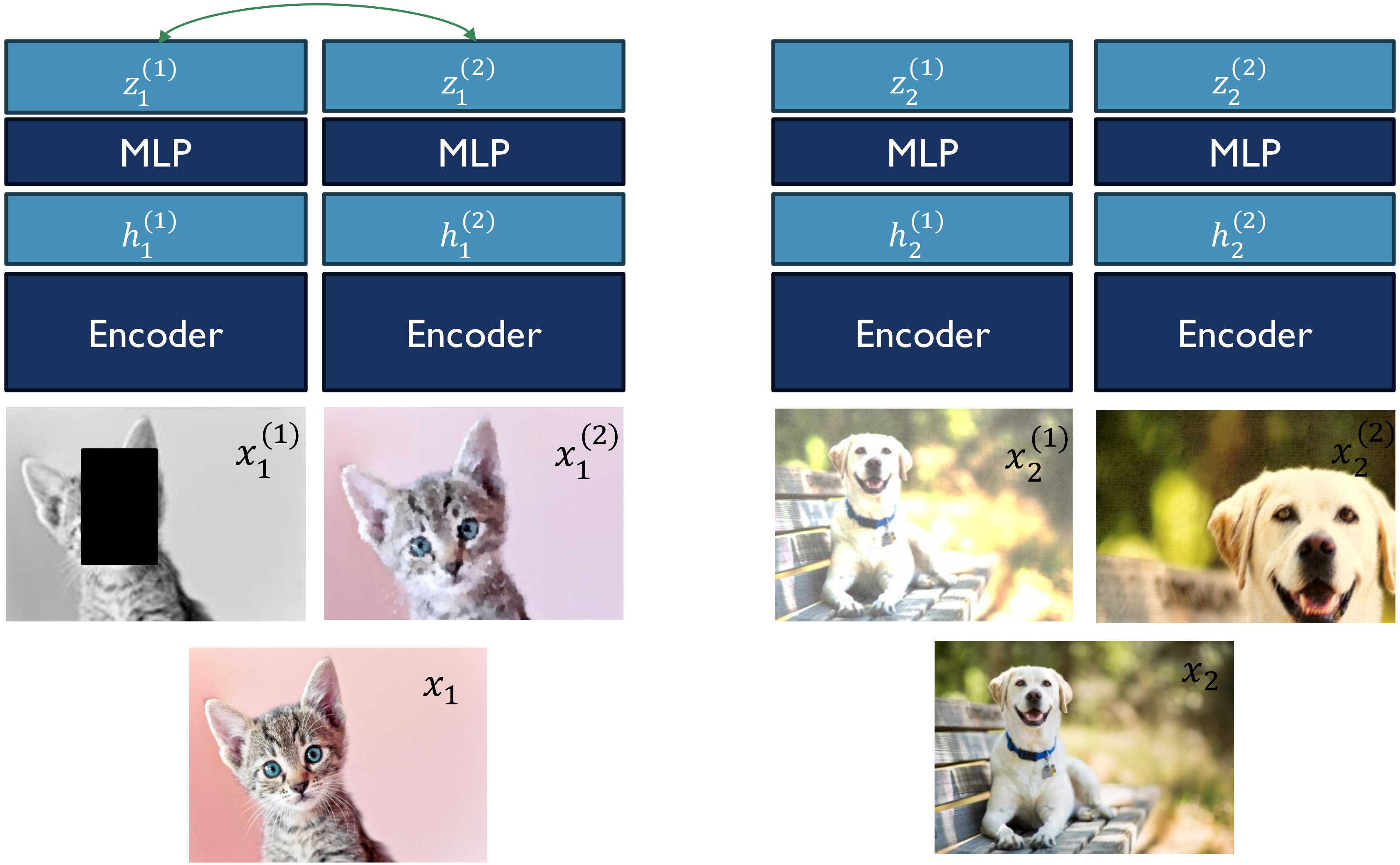
MLP
g(.)

z

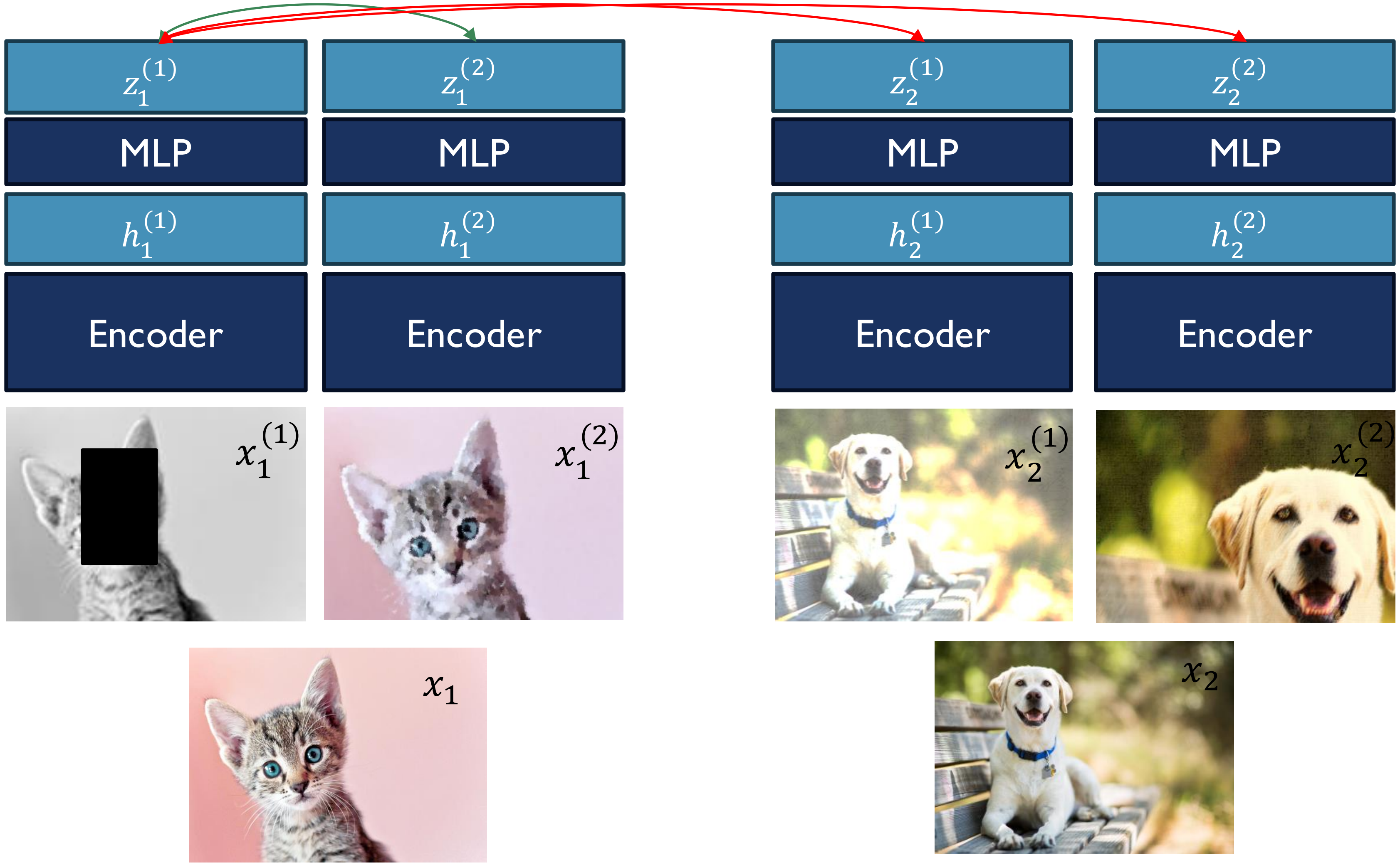
h = The representation used for downstream tasks (e.g. classification)

z = A smaller representation used to compute the contrastive loss

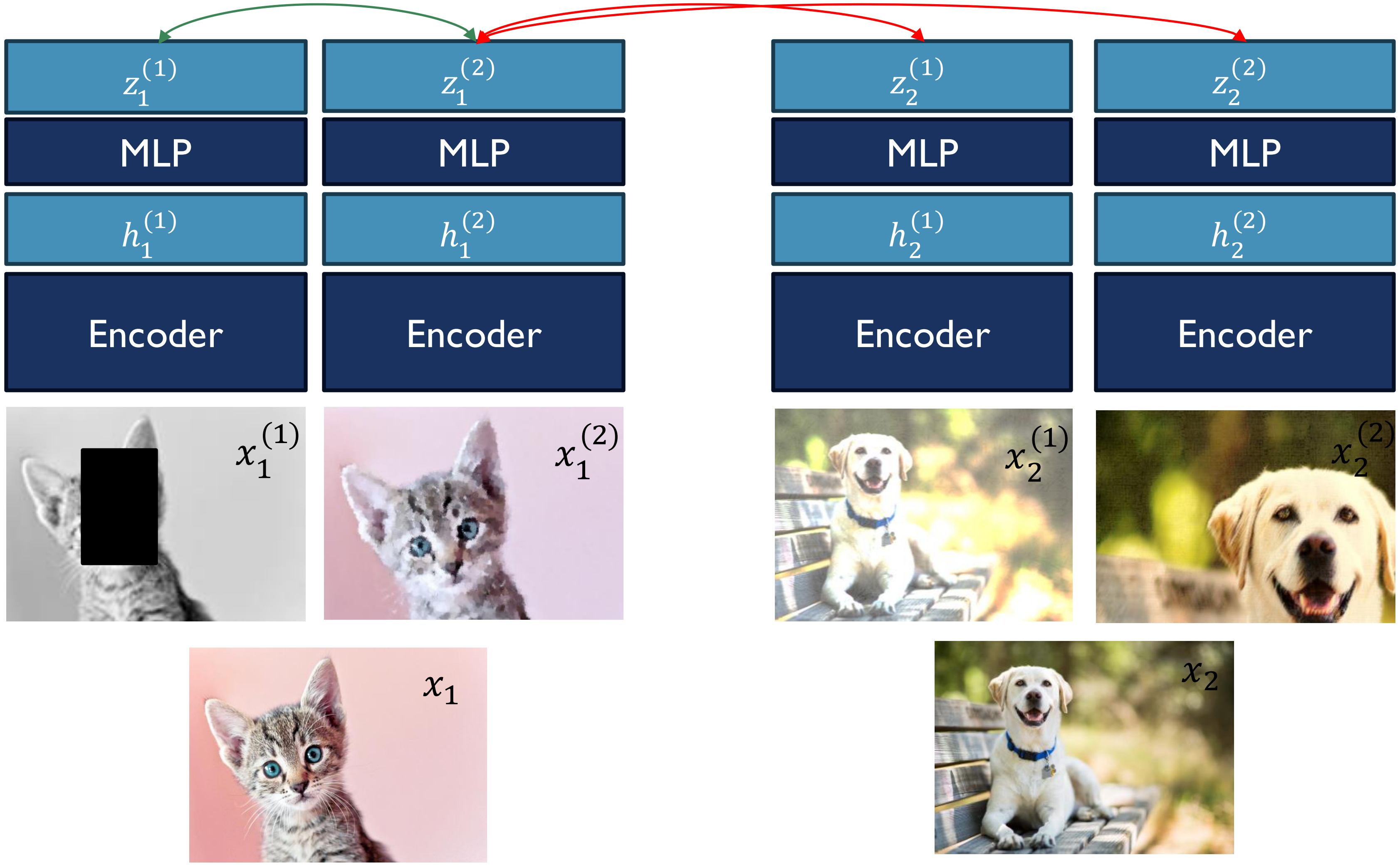
CONTRASTIVE LEARNING – MAIN COMPONENTS



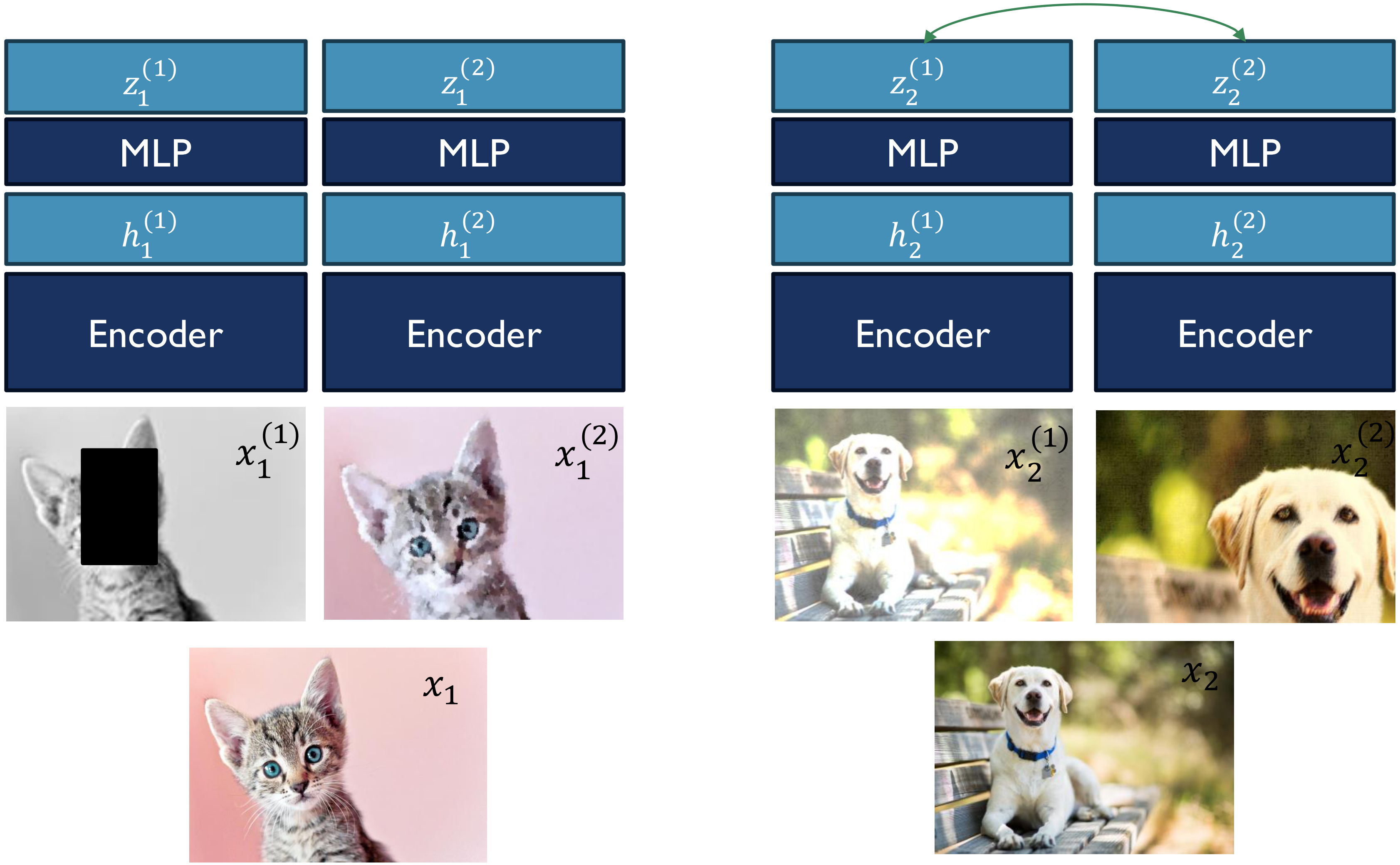
CONTRASTIVE LEARNING – MAIN COMPONENTS



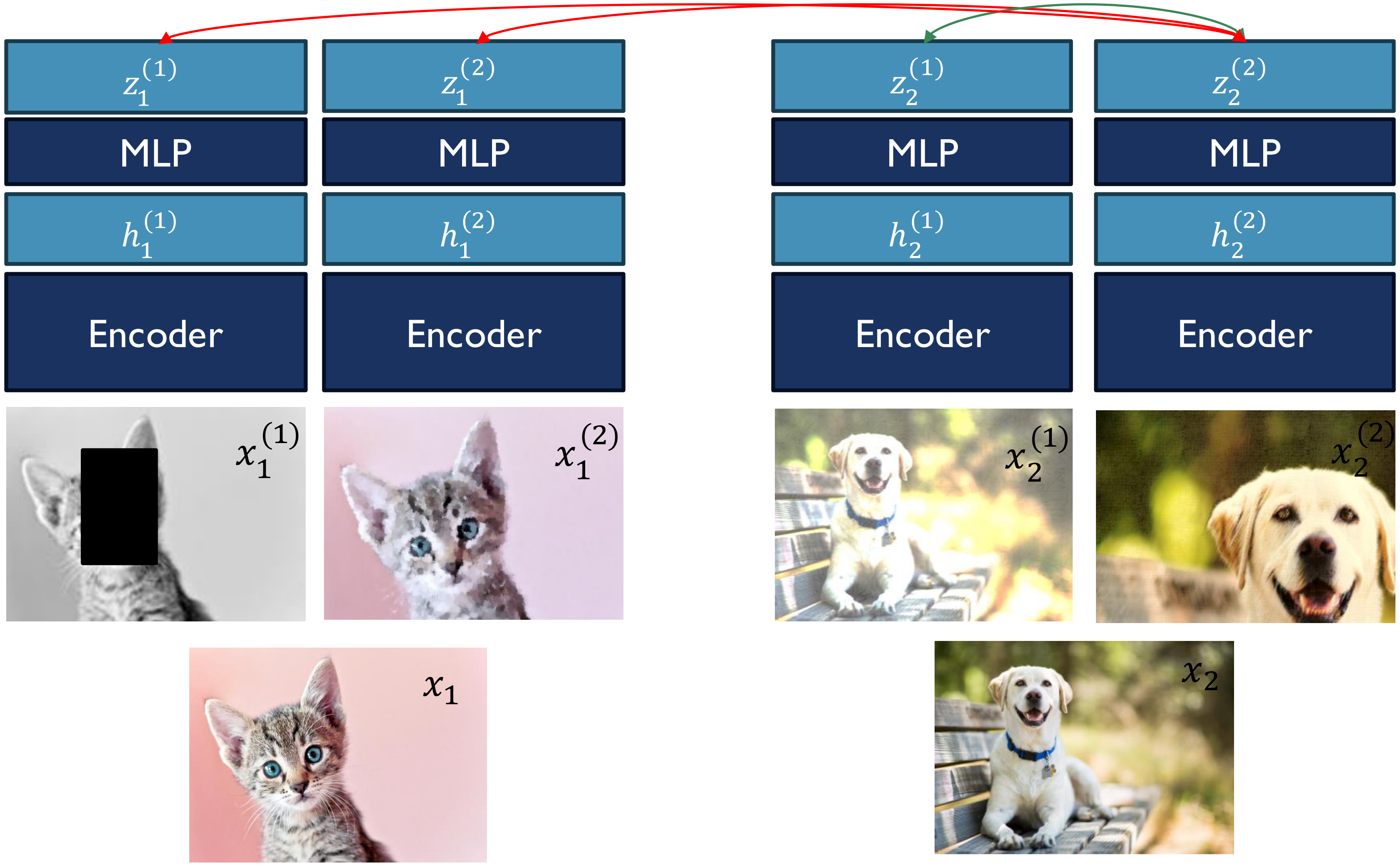
CONTRASTIVE LEARNING – MAIN COMPONENTS



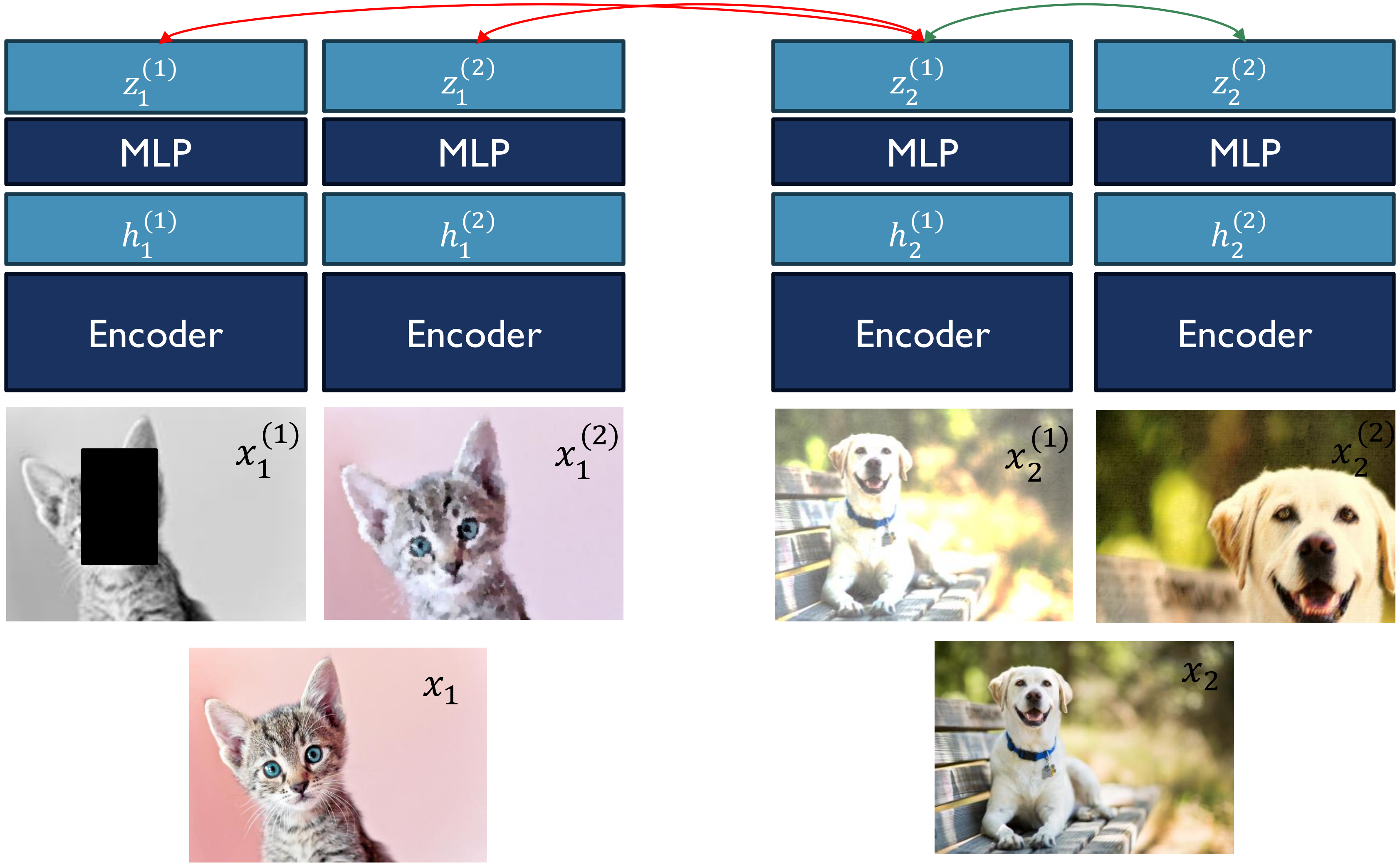
CONTRASTIVE LEARNING – MAIN COMPONENTS



CONTRASTIVE LEARNING – MAIN COMPONENTS



CONTRASTIVE LEARNING – MAIN COMPONENTS



ONE KEY INGREDIENT TO CONTRASTIVE LEARNING: THE AUGMENTATION PIPELINE

Augmentation pipeline

- Needs to reflect **what information the model should disregard** and **what it should focus on**
- **Needs to be hard enough**, otherwise trivial information is learned

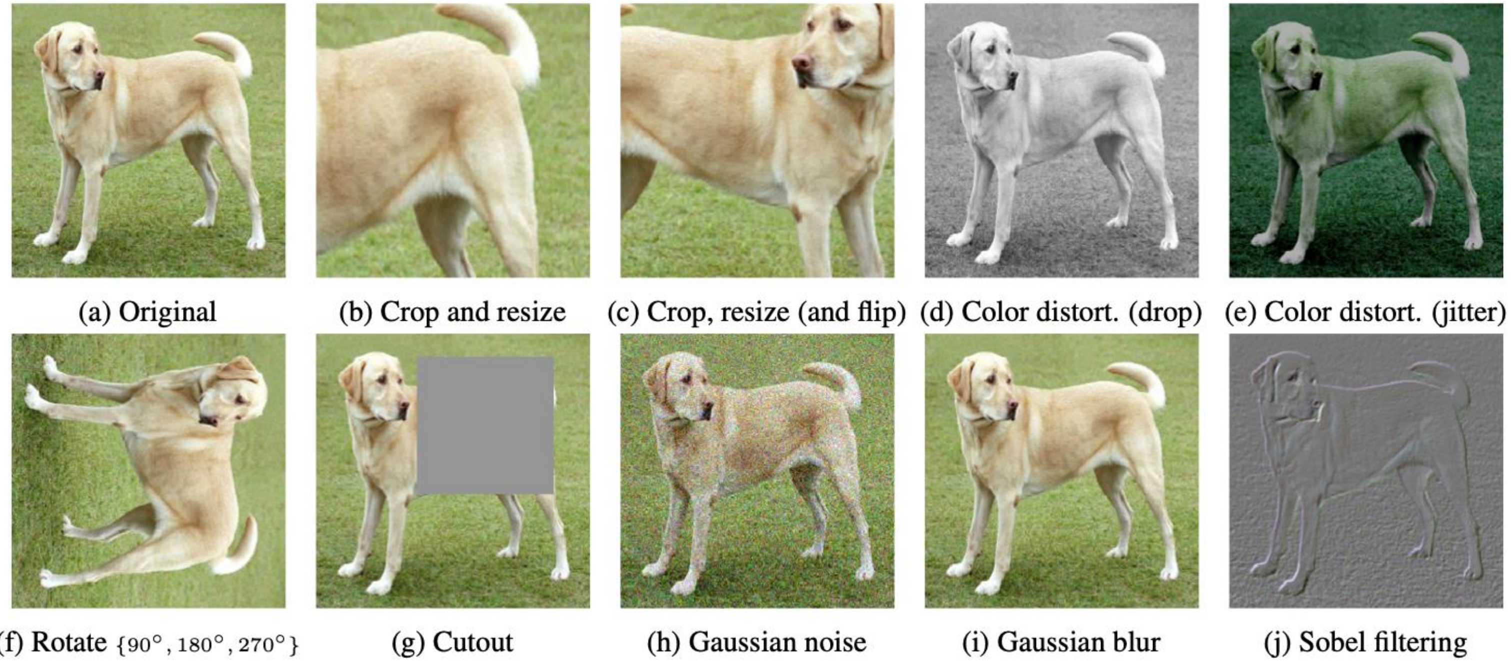


Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we *only* test these operators in ablation, the *augmentation policy used to train our models* only includes *random crop (with flip and resize)*, *color distortion*, and *Gaussian blur*. (Original image cc-by: Von.grzanka)

Source: Chen, Ting, et al. "A simple framework for contrastive learning of visual representations."

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- What does the SimCLR loss function look like ?
- How can we attract positive pairs while repelling negative pairs?

→ In the next slide we dive into the **NT-Xent loss** (normalised temperature contrastive loss) proposed in the SimCLR courseowkr.

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- How do we measure similarity ?

$$\boxed{\mathrm{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\|}$$

Scalar product of normalised embeddings.

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- In SimCLR, for each positive pair (i, j) we have the following loss:

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- In SimCLR, for each positive pair (i, j) we have the following loss:

$$\ell_{i,j} = -\boxed{\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}},$$

To mininimize the loss → Maximize this quantity

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- In SimCLR, for each positive pair (i, j) we have the following loss:

$$\ell_{i,j} = -\boxed{\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}},$$

To mininimize the loss → Maximize this quantity
→ Maximise the numerator
→ Minimise the denominator

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- In SimCLR, for each positive pair (i, j) we have the following loss:

$$\ell_{i,j} = -\log \frac{\boxed{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$

- Intuitively, to decrease the loss:
  - the network needs to **pull the positive pairs closer** (maximise the numerator)
  -

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- In SimCLR, for each positive pair (i, j) we have the following loss:

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\boxed{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}},$$

- Intuitively, to decrease the loss:
  - the network needs to **pull the positive pairs closer** (maximise the numerator)
  - While **pushing negative pairs far from each other** (minimise the denominator)

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- In SimCLR, for each positive pair:

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$

- Temperature τ controls how much to penalise hard negatives (negative pairs wrongly mapped closed to each other). A low temperature penalise them more.

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- In SimCLR, for each positive pair:

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)} ,$$

- The final loss is computed across all positive pairs, both (i,j) and (j,i), in a mini-batch, by averaging all $l_{i,j}$.

CONTRASTIVE LEARNING – SIMCLR FULL ALGORITHM

Concretely to build the loss for the entire batch with N samples one needs to follow the following process:

1. **Compute all projected embeddings for the first view** of each sample $\boldsymbol{z}^{(1)} = [z_1^{(1)}; z_2^{(1)}; \ldots; z_N^{(1)}]$, vector of size [N, feature_dim]. For that first apply augmentation, pass through encoder E, finally pass through MLP projector to get z.
2. **Repeat it for the second view** $\boldsymbol{z}^{(2)} = [z_1^{(2)}; z_2^{(2)}; \ldots; z_N^{(2)}]$
3. Then **gather all representations in one big vector** (to compute the similarities for the denominator), of size [2N, feature_dim]

$$\boldsymbol{z} = \left[\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}\right] = [z_1^{(1)}; \ldots; z_N^{(1)}; z_1^{(2)}; \ldots; z_N^{(2)}]$$

4. In this vector the positive pairs will be elements i and $i + N$ (and the opposite)

Hence the **final loss averaged over all positive pairs** will be:

$$\mathrm{L}_{Batch} = \frac{1}{2N} \sum_{i=1}^{N} \left[l_{i,i+N} + l_{i+N,i}\right]$$

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- There exists also other loss functions, for example: the triplet loss

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \boxed{\|f(\mathbf{x}) - f(\mathbf{x}^+)\|_2^2} - \boxed{\|f(\mathbf{x}) - f(\mathbf{x}^-)\|_2^2} + \epsilon\right)$$

- Where x is one image, x^+ is the corresponding positive and x^- a randomly selected negative image. ε is the margin parameter controlling how far the negatives should be compared to the positive pairs (temperature τ plays this role in NT-Xent loss).

Many more losses have been proposed, see https://lilianweng.github.io/posts/2021-05-31-contrastive/#contrastive-loss

CONTRASTIVE LEARNING - LOSS FUNCTIONS

- There exists also other loss functions, for example: the triplet loss

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \boxed{\|f(\mathbf{x}) - f(\mathbf{x}^+)\|_2^2} - \boxed{\|f(\mathbf{x}) - f(\mathbf{x}^-)\|_2^2} + \epsilon\right)$$

If the distance between $f(x)$ and $f(x^+)$ is smaller
than the distance between $f(x)$ and $f(x^-)$ plus the margin,
this qty is negative so the loss is zero.

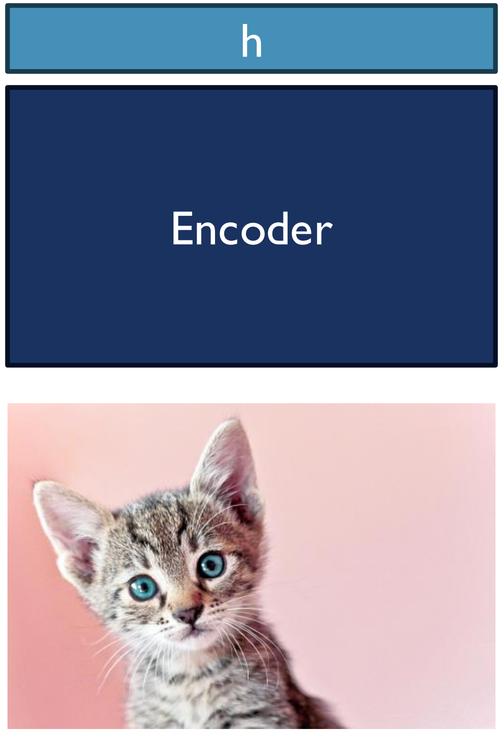
- Where x is one image, x^+ is the corresponding positive and x^- a randomly selected negative image. ε is the margin parameter between positives and negatives.

Many more losses have been proposed, see https://lilianweng.github.io/posts/2021-05-31-contrastive/#contrastive-loss

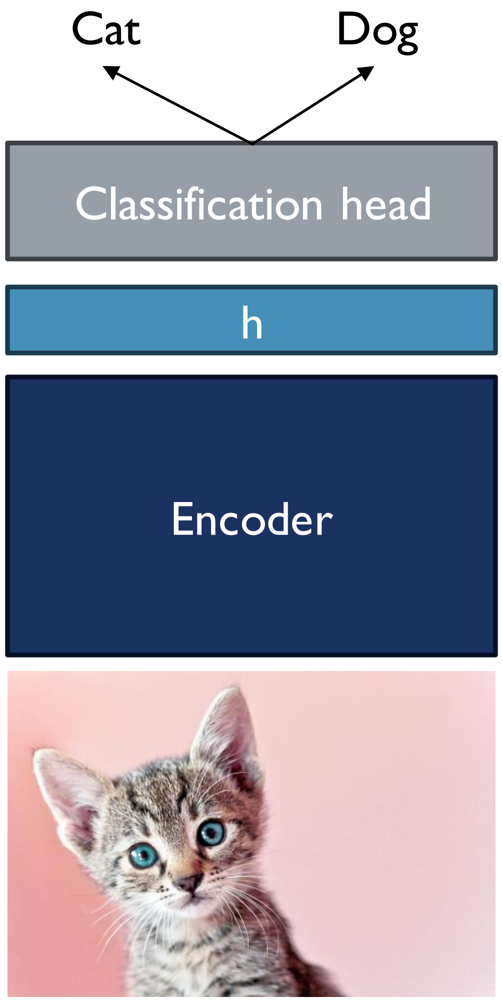
EVALUATION OF SELF-SUPERVISED

- Once we trained a SSL model, how can use it for downstream tasks?
- **How can we check if the representation learned are useful ?**

EVALUATION OF SELF-SUPERVISED MODELS: FINETUNING

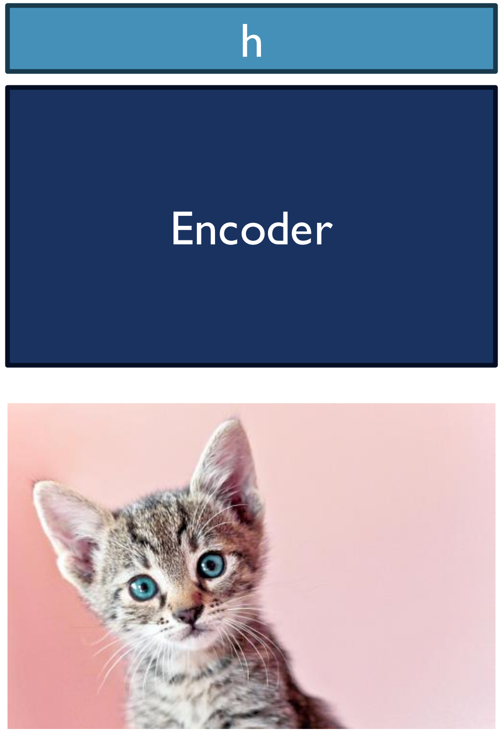


Step 1: pretrain the encoder with self-supervised learning e.g. SimCLR

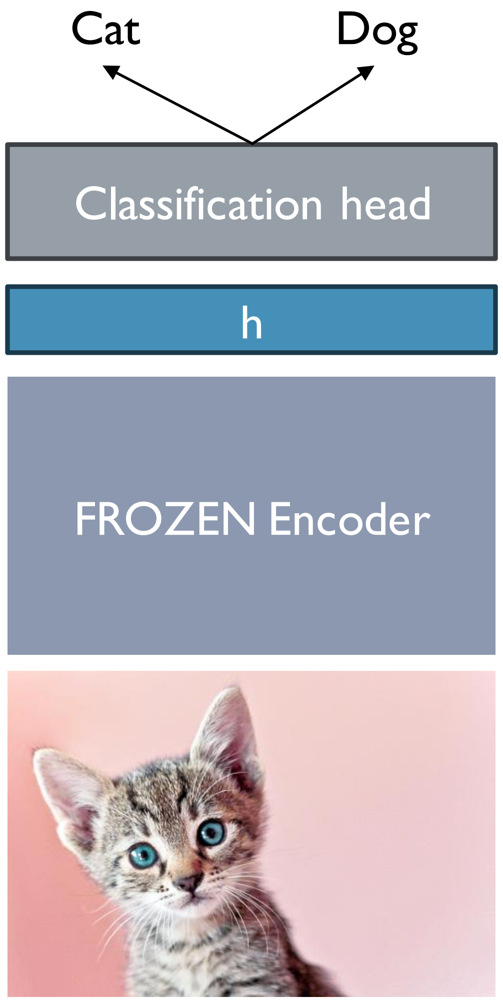


Step 2: load the pretrained encoder as starting weights to train a classifier with cross-entropy loss.

EVALUATION OF SELF-SUPERVISED MODELS: LINEAR PROBING (AKA LINEAR EVALUATION)



Step I: pretrain the encoder with self-supervised learning e.g. SimCLR



Step 2: load the pretrained encoder, **freeze all the weights of the encoder**. Train a classification head on top of the frozen encoder with cross-entropy loss.

BATCH SIZE: ANOTHER KEY INGREDIENT TO SIMCLR

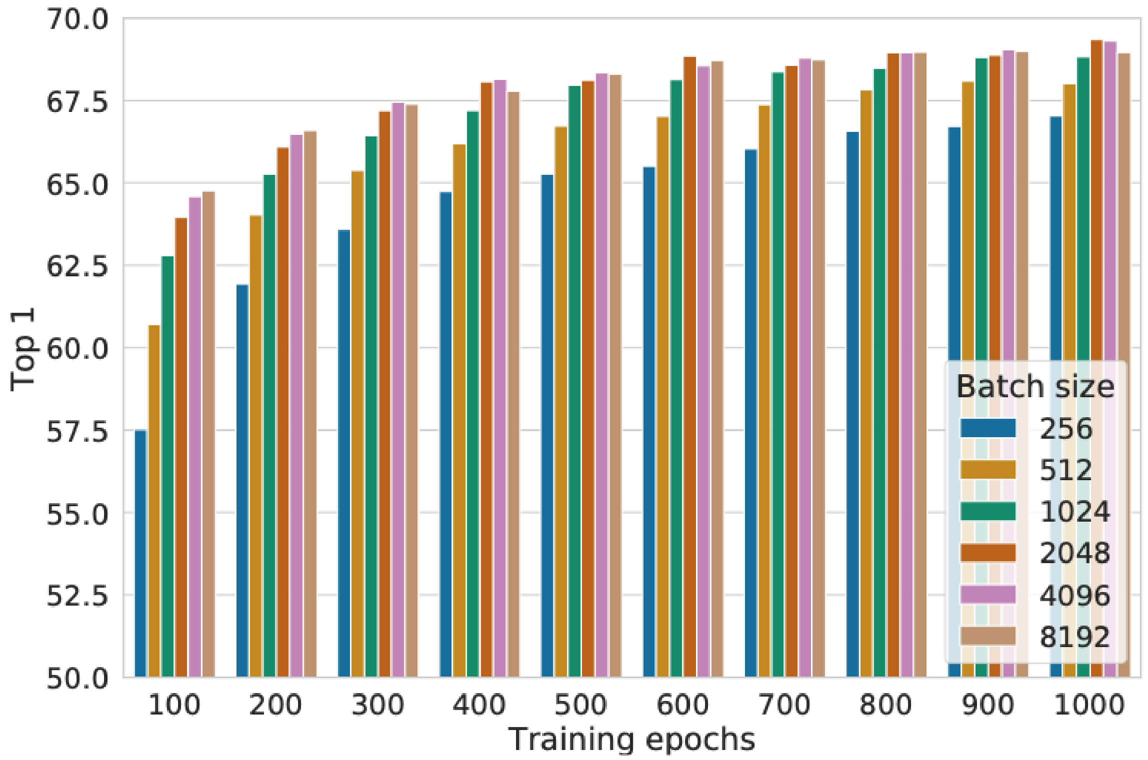


Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

To learn good representation, it is crucial to **use very large batch sizes with SimCLR**

Source: Chen, Ting, et al. "A simple framework for contrastive learning of visual representations."

FOLLOW-UP WORKS ON SIMCLR

- SimCLR requires very large batch size to provide good representation → this means very high computational requirements (expensive, difficult to train)
- Others have come up with alternatives to decrease the requirement on the batch size
- One example is **Bootstrap Your Own Latent**

But they are many more, pointers can be found at https://encord.com/blog/guide-to-contrastive-learning/#h5

Grill, Jean-Bastien, et al. "Bootstrap your own latent-a new approach to self-supervised learning." *Advances in neural information processing systems* 33 (2020): 21271-21284.

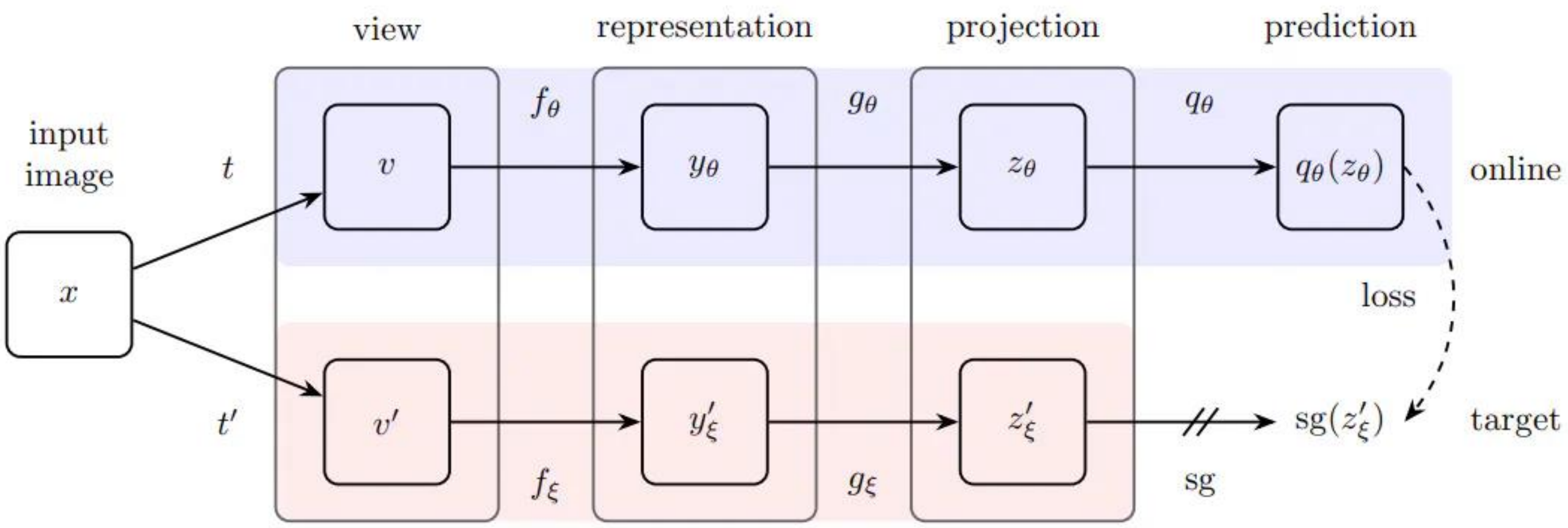
BOOTSTRAP YOUR OWN LATENT (BYOL)

BYOL removes the need to use negative pairs in the loss functions, instead one just optimises the similarity on the positive pairs → less need of big batch sizes

For that, they needed to introduce a new architecture to avoid learning trivial embeddings (otherwise all converge to the same point).

Grill, Jean-Bastien, et al. "Bootstrap your own latent-a new approach to self-supervised learning." *Advances in neural information processing systems* 33 (2020): 21271-21284. https://arxiv.org/abs/2006.07733

BYOL

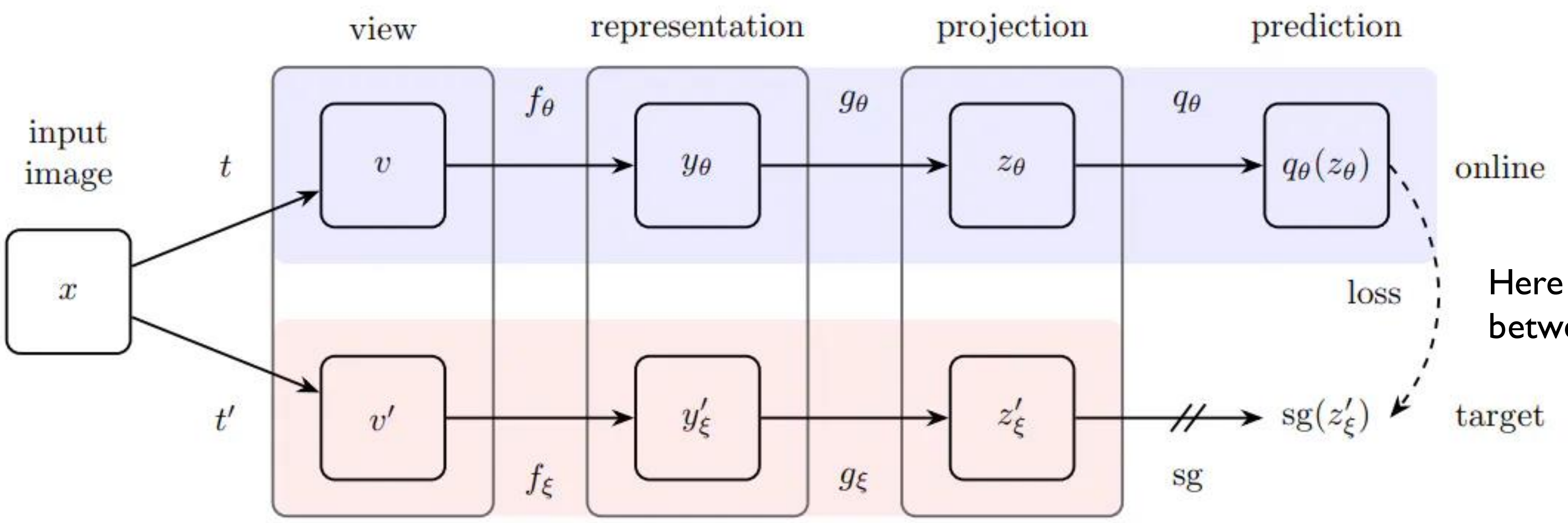


"Student network" weights learned by gradient descent.

"Teacher network" weights are a moving average of the weights of the student network.

https://arxiv.org/abs/2006.07733

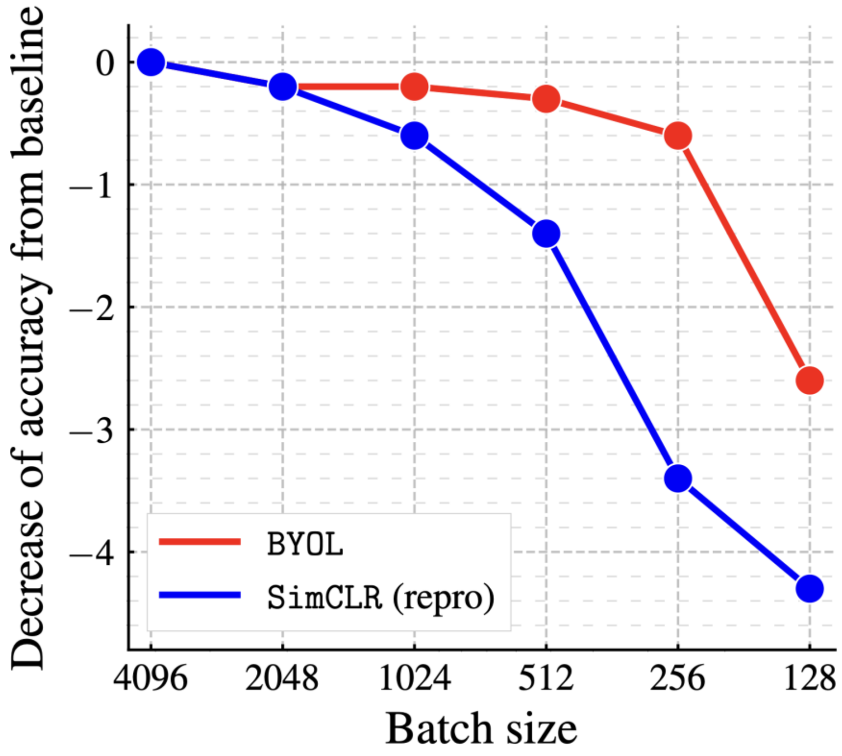
BYOL



Here the loss maximises the similarity between positive pairs embeddings.

https://arxiv.org/abs/2006.07733

BYOL



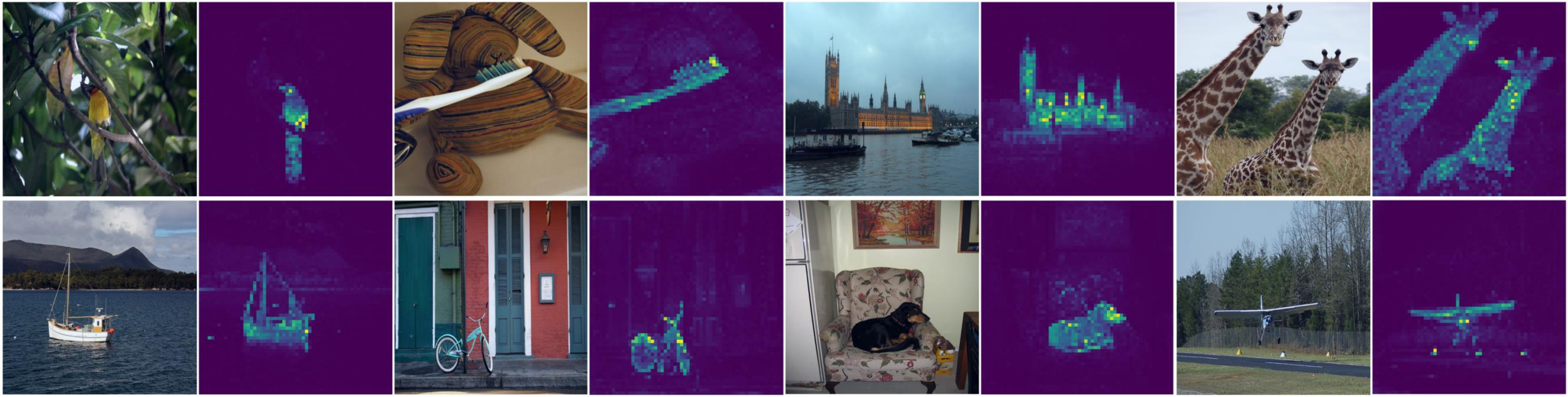
BYOL is **more robust to smaller batch size** than SimCLR.

From https://arxiv.org/abs/2006.07733

SELF-SUPERVISED SEGMENTATION WITH DINO

EMERGING PROPERTIES IN SELF-SUPERVISED VISION TRANSFORMERS, META AI

- Very similar to training BYOL, but using vision transformers as encoders.
- Thanks to the ViT architecture, one can visualize the attention maps of the network for generating self—supervised segmentation maps



https://arxiv.org/pdf/2104.14294.pdf

SELF-SUPERVISED SEGMENTATION WITH DINO

EMERGING PROPERTIES IN SELF-SUPERVISED VISION TRANSFORMERS, META AI

We can also **visualise the learned embeddings during training** to inspect the learned representation.

Embeddings are projected to a 2D space for visualisation.

From official blog

SELF-SUPERVISED SEGMENTATION WITH DINO

All the details: [official blog](), [paper](), [code]()

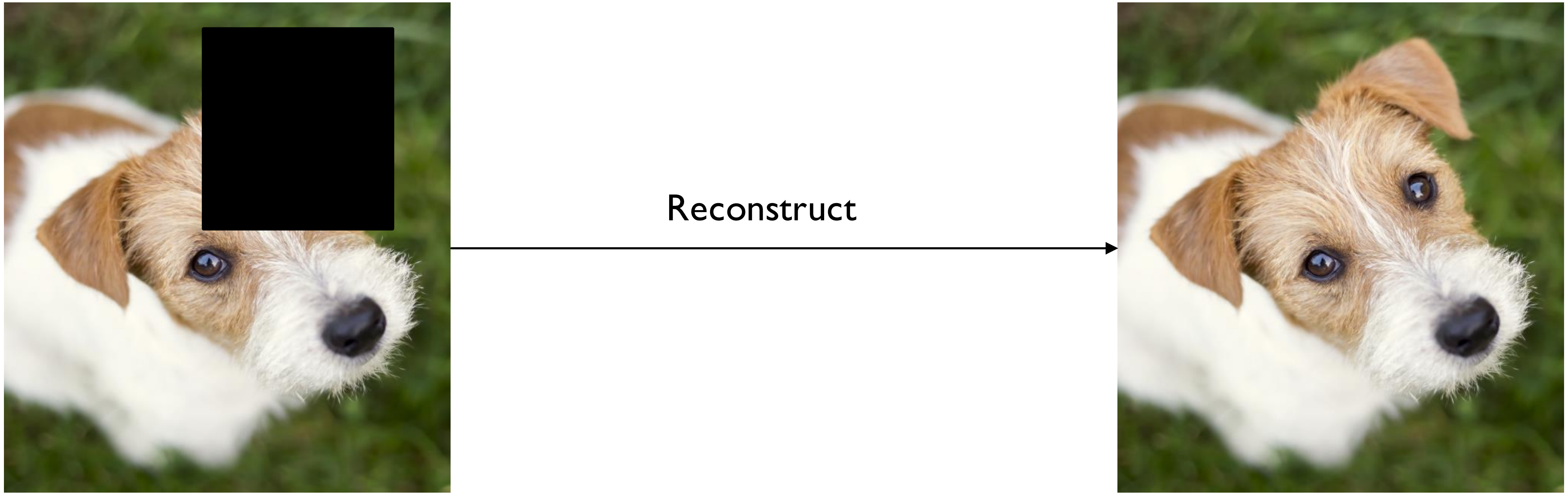
3. GENERATIVE APPROACHES TO SELF-SUPERVISED LEARNING

GENERATIVE APPROACHES TO SELF-SUPERVISED LEARNING

- Contrastive learning is one pretext task that has proven very useful in self-supervised learning → but it's not the only one !
- In particular, CL has some drawbacks: large batch sizes, design of the augmentation pipeline etc.

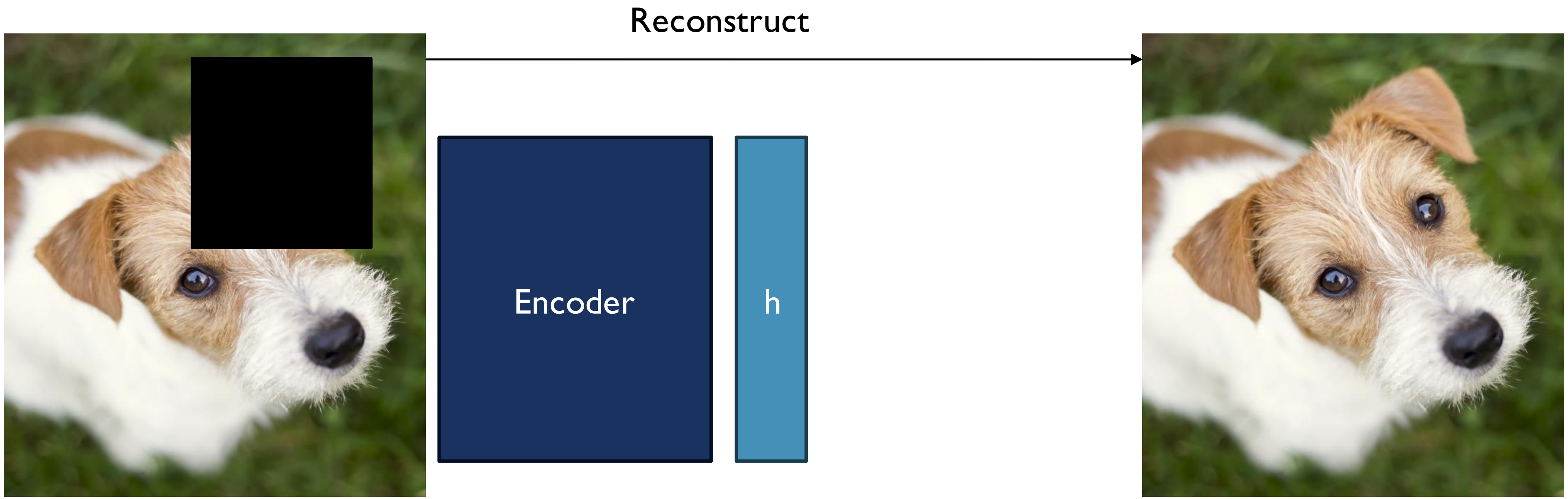
GENERATIVE APPROACHES TO SELF-SUPERVISED LEARNING

- Another way to use the image as supervision signal is **to teach the network to reconstruct the original image from a masked image.**



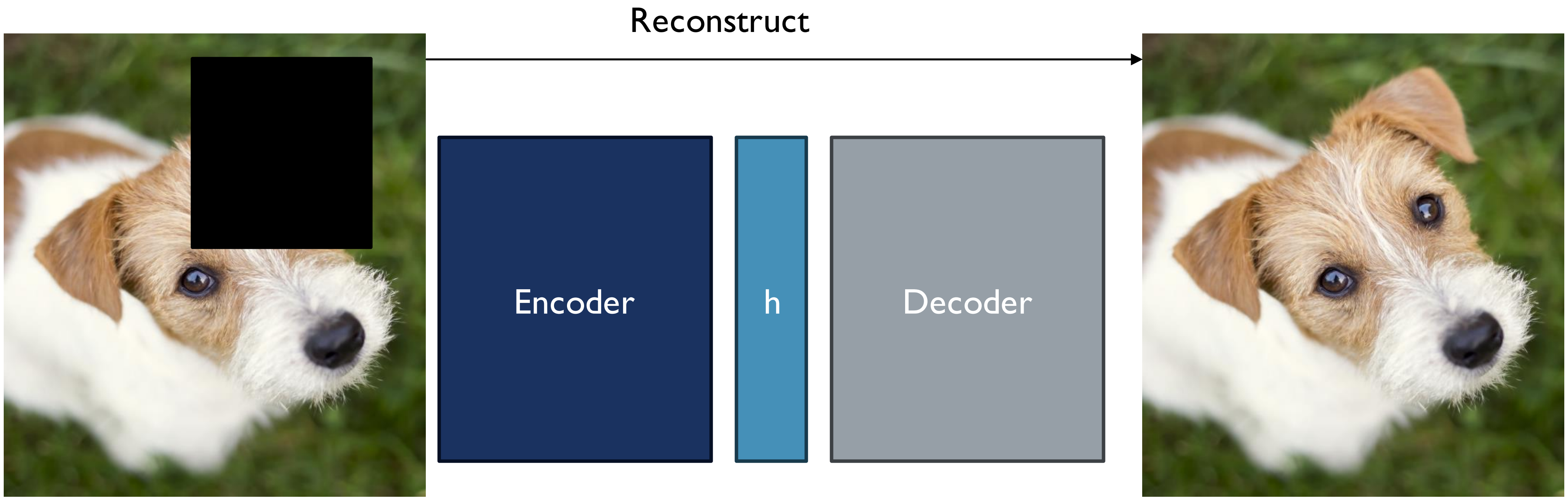
MASKED AUTO-ENCODERS ANOTHER APPROACH TO SELF-SUPERVISED LEARNING

- Instead of recognising pairs of images, learn how to fill in the gaps.

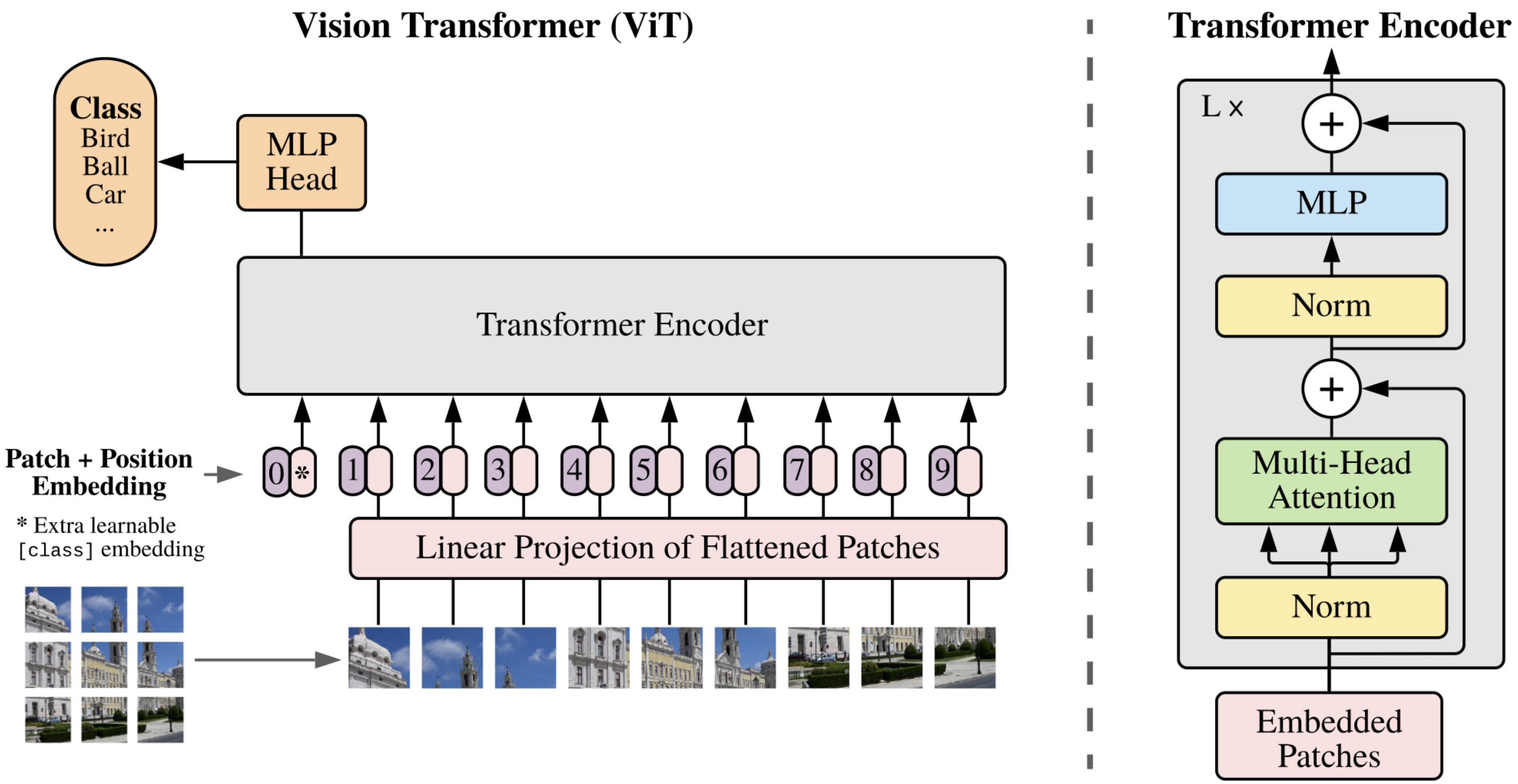


MASKED AUTO-ENCODERS ANOTHER APPROACH TO SELF-SUPERVISED LEARNING

- Instead of recognising pairs of images, learn how to fill in the gaps.



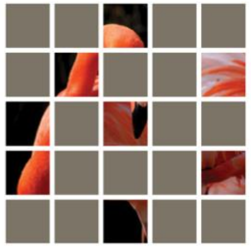
LITTLE REFRESHER ON VISION TRANSFORMERS



https://arxiv.org/pdf/2010.11929v2.pdf

MASKED AUTO ENCODERS ARE SCALABLE VISION LEARNERS – HE ET AL. CVPR 22'

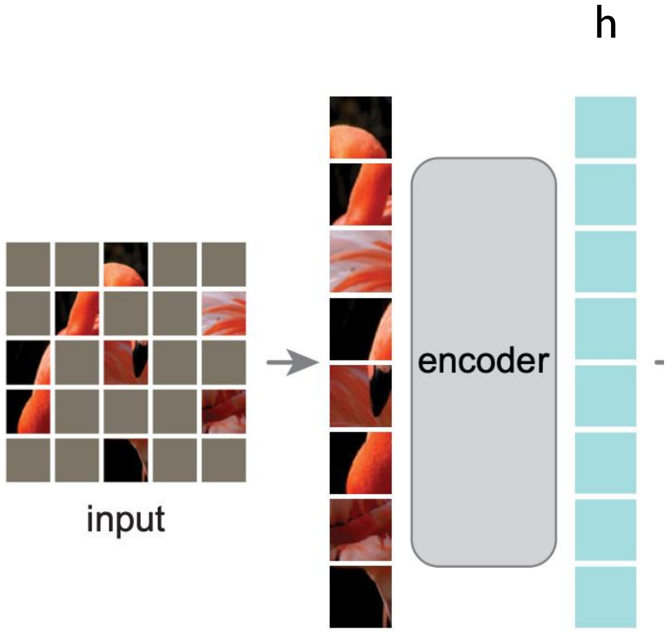
- Using patch-wise encoders and decoders



From https://openaccess.thecvf.com/content/CVPR2022/papers/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.pdf

MASKED AUTO ENCODERS ARE SCALABLE VISION LEARNERS – HE ET AL. CVPR 22'

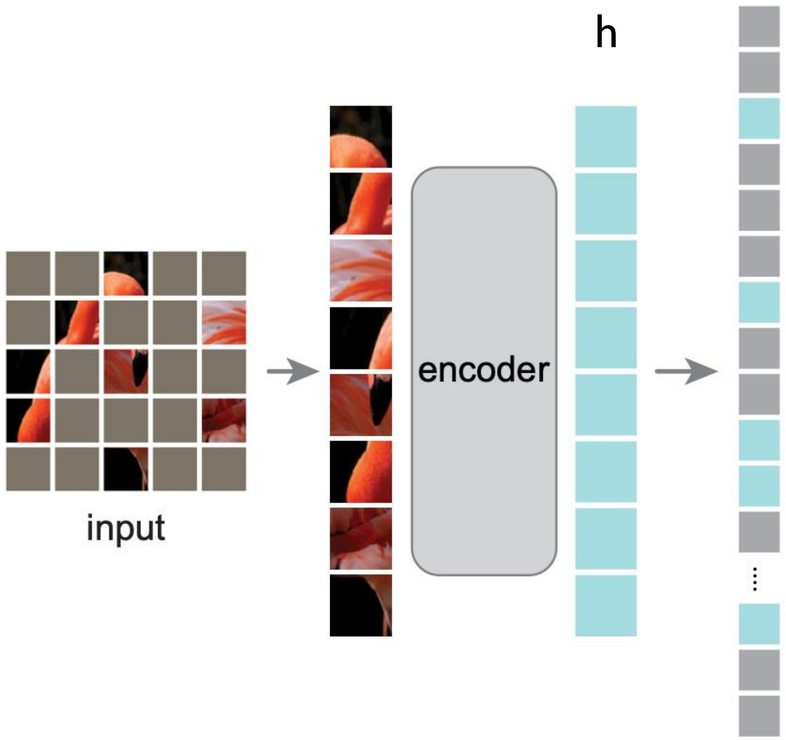
- Using patch-wise encoders and decoders



https://openaccess.thecvf.com/content/CVPR2022/papers/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.pdf

MASKED AUTO ENCODERS ARE SCALABLE VISION LEARNERS – HE ET AL. CVPR 22'

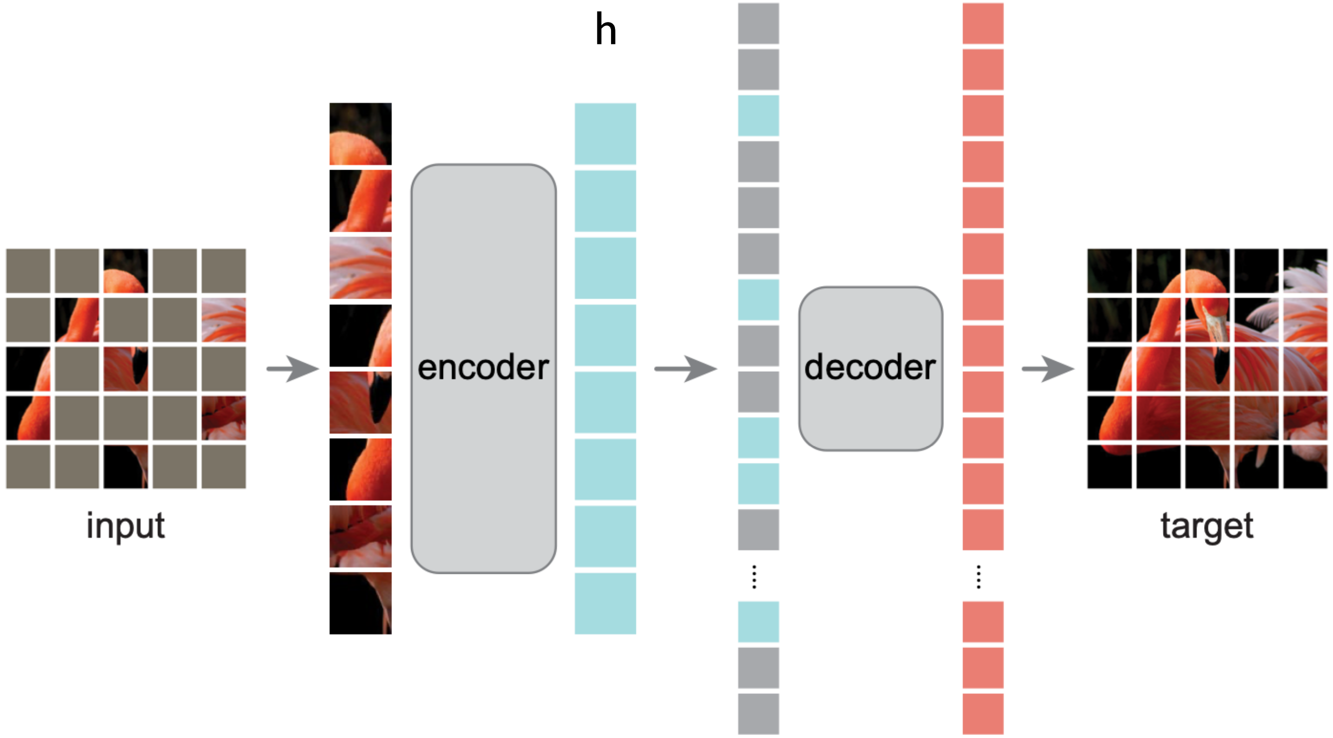
- Using patch-wise encoders and decoders



https://openaccess.thecvf.com/content/CVPR2022/papers/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.pdf

MASKED AUTO ENCODERS ARE SCALABLE VISION LEARNERS – HE ET AL. CVPR 22'

- Using patch-wise encoders and decoders



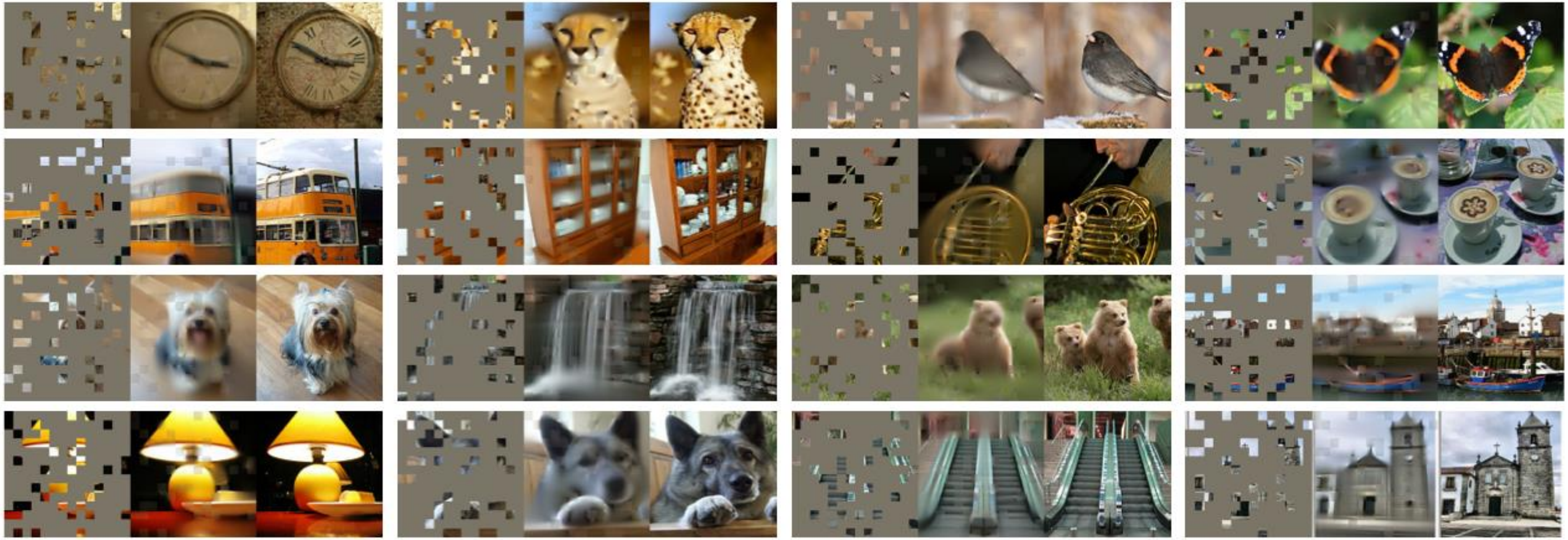
https://openaccess.thecvf.com/content/CVPR2022/papers/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.pdf

MASKED AUTO ENCODERS ARE SCALABLE VISION LEARNERS

- Loss function is the mean squared error between predicted $\hat{x}$ and the real image x

$$MSE = \sum(\hat{x}_i - x_i)^2$$, where i is the pixel index.

MASKED AUTO ENCODERS ARE SCALABLE VISION LEARNERS



From https://openaccess.thecvf.com/content/CVPR2022/papers/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.pdf

MASKED AUTO ENCODERS ARE SCALABLE VISION LEARNERS

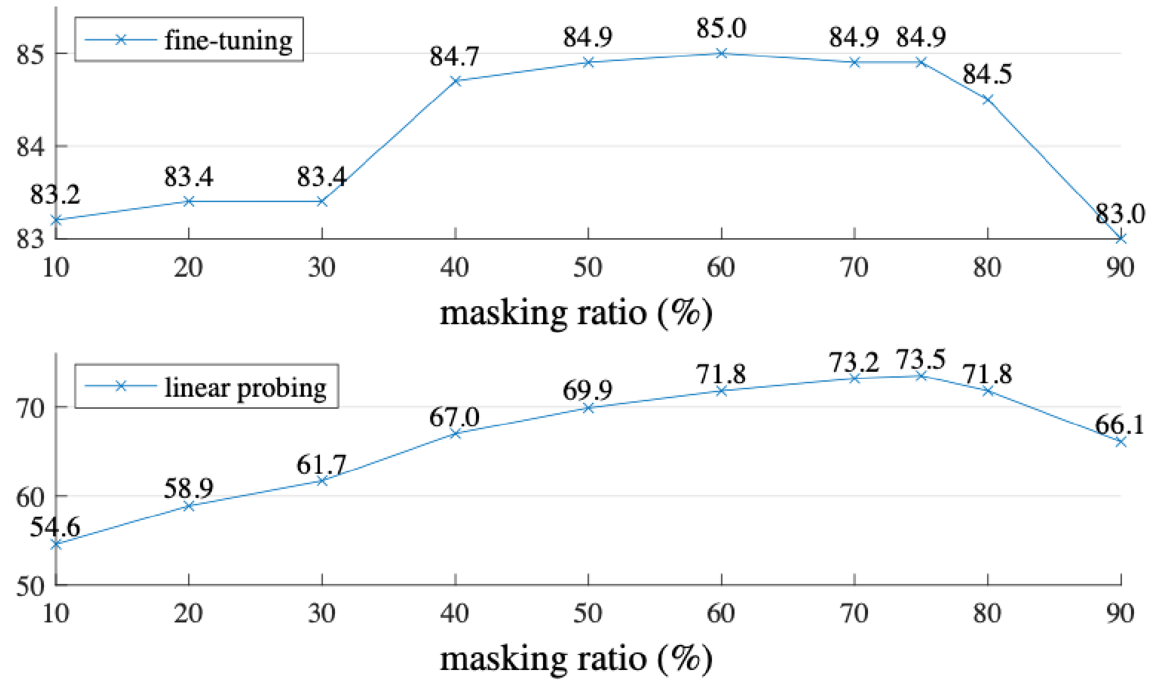


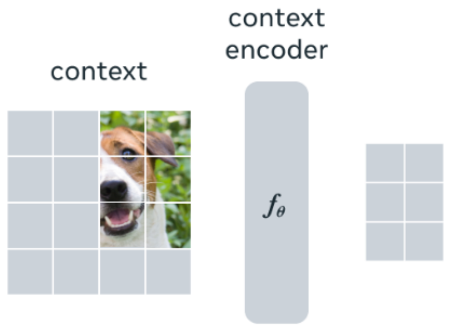
Figure 5. **Masking ratio**. A high masking ratio (75%) works well for both fine-tuning (top) and linear probing (bottom). The y-axes are ImageNet-1K validation accuracy (%) in all plots in this paper.

From https://openaccess.thecvf.com/content/CVPR2022/papers/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.pdf

MIX OF CONTRASTIVE AND MAE: I-JEPA

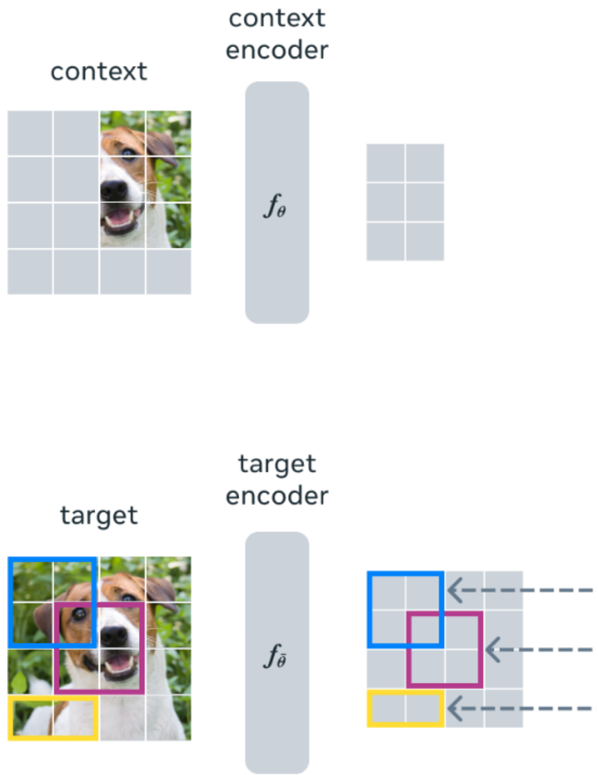
- MAE avoids some of the drawbacks of CL, however training **a full reconstruction model is quite expensive**.
- Maybe it is not necessarily to fully learn a perfect decoder model since we throw it away after training ?
- → The idea behind i-Jepa: combining strength of CL and MAE approaches.

From https://ai.meta.com/blog/yann-lecun-ai-model-i-jepa/



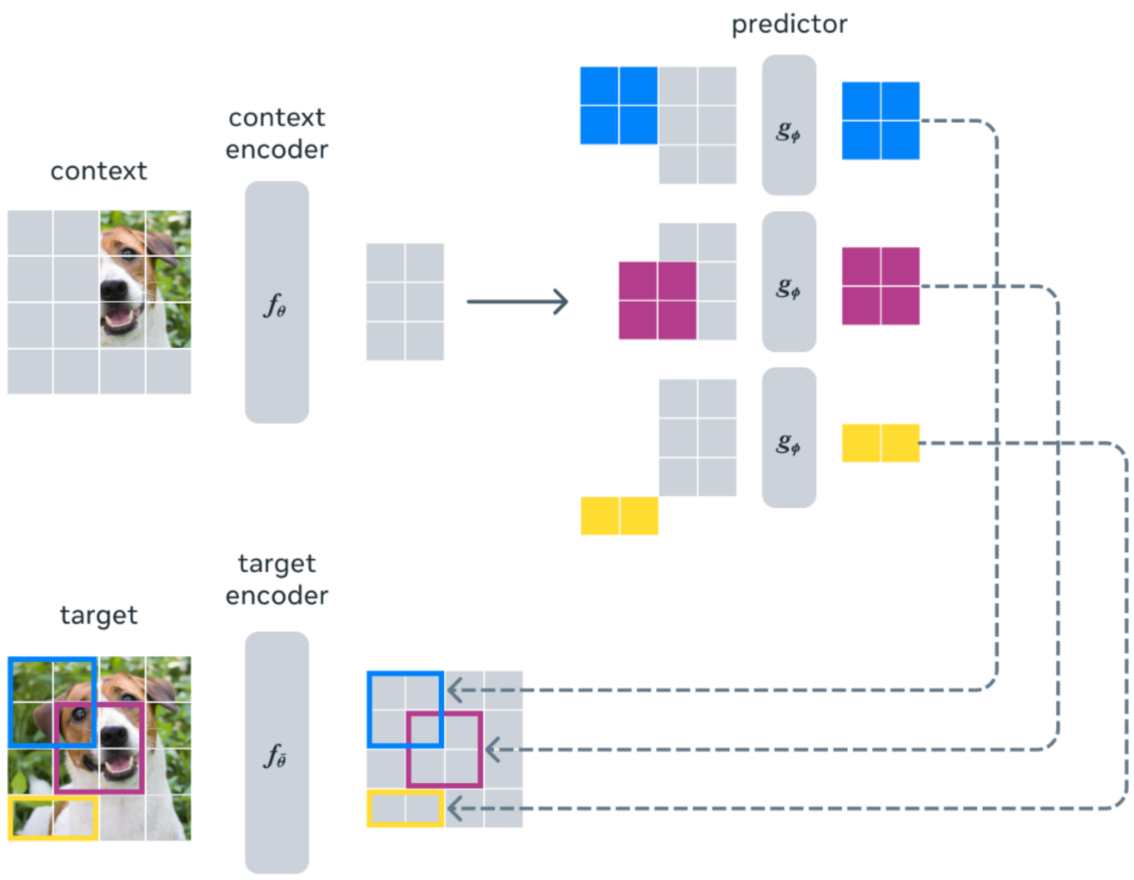
The Image-based Joint-Embedding Predictive Architecture (I-JEPA) uses a single context block to predict the representations of various target blocks originating from the same image. The context encoder is a Vision Transformer (ViT) that only processes the visible context patches. The predictor is a narrow ViT that takes the context encoder output and predicts the representations of a target block at a specific location, conditioned on positional tokens of the target (shown in color). The target representations correspond to the outputs of the target-encoder, the weights of which are updated at each iteration via an exponential moving average of the context encoder weights.

https://ai.meta.com/blog/yann-lecun-ai-model-i-jepa/



The Image-based Joint-Embedding Predictive Architecture (I-JEPA) uses a single context block to predict the representations of various target blocks originating from the same image. The context encoder is a Vision Transformer (ViT) that only processes the visible context patches. The predictor is a narrow ViT that takes the context encoder output and predicts the representations of a target block at a specific location, conditioned on positional tokens of the target (shown in color). The target representations correspond to the outputs of the target-encoder, the weights of which are updated at each iteration via an exponential moving average of the context encoder weights.

https://ai.meta.com/blog/yann-lecun-ai-model-i-jepa/



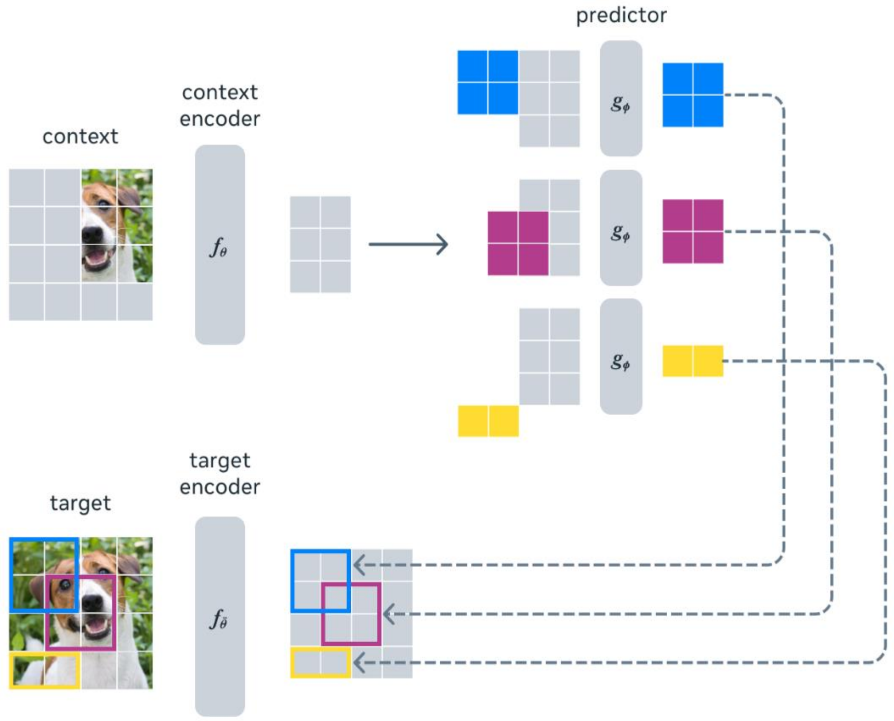
The Image-based Joint-Embedding Predictive Architecture (I-JEPA) uses a single context block to predict the representations of various target blocks originating from the same image. The context encoder is a Vision Transformer (ViT) that only processes the visible context patches. The predictor is a narrow ViT that takes the context encoder output and predicts the representations of a target block at a specific location, conditioned on positional tokens of the target (shown in color). The target representations correspond to the outputs of the target-encoder, the weights of which are updated at each iteration via an exponential moving average of the context encoder weights.

https://ai.meta.com/blog/yann-lecun-ai-model-i-jepa/

MIX OF CL AND MAE: I-JEPA

Loss. The loss is simply the average L_2 distance between the predicted patch-level representations $\hat{\boldsymbol{s}}_y(i)$ and the target patch-level representation $\boldsymbol{s}_y(i)$; i.e.,

$$\frac{1}{M}\sum_{i=1}^{M} D\left(\hat{\boldsymbol{s}}_y(i), \boldsymbol{s}_y(i)\right) = \frac{1}{M}\sum_{i=1}^{M}\sum_{j\in B_i} \|\hat{\boldsymbol{s}}_{y_j} - \boldsymbol{s}_{y_j}\|_2^2.$$

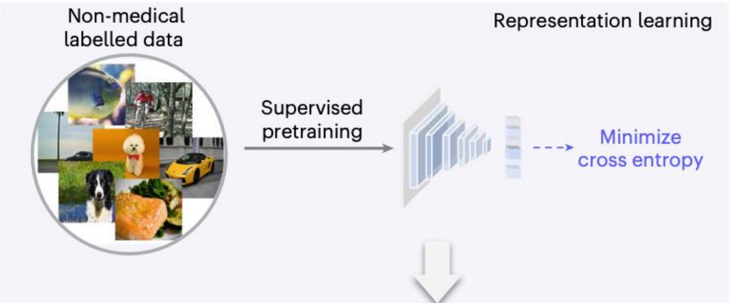


MIX OF CL AND MAE: I-JEPA

https://arxiv.org/pdf/2301.08243.pdf

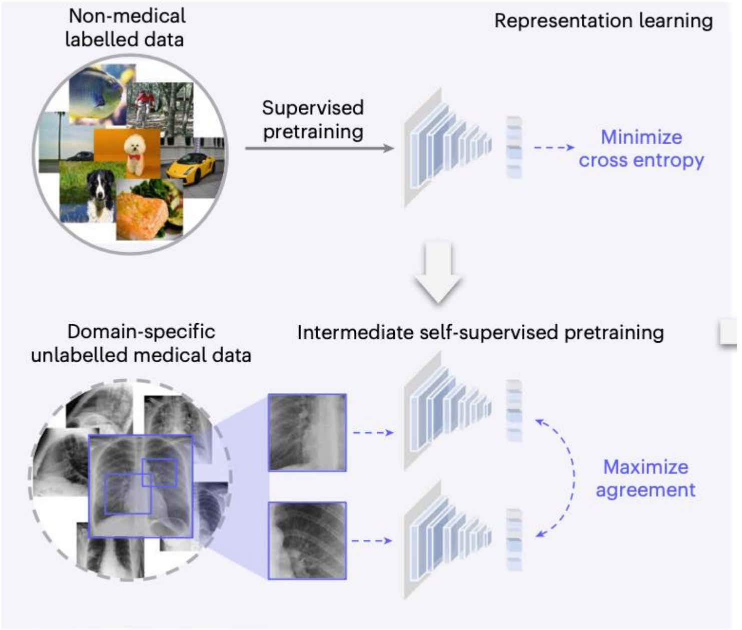
EXAMPLES OF APPLICATIONS OF SSL MODELS IN HEALTHCARE

SUCCESSFUL APPLICATIONS OF SELF-SUPERVISED MODELS FOR HEALTHCARE



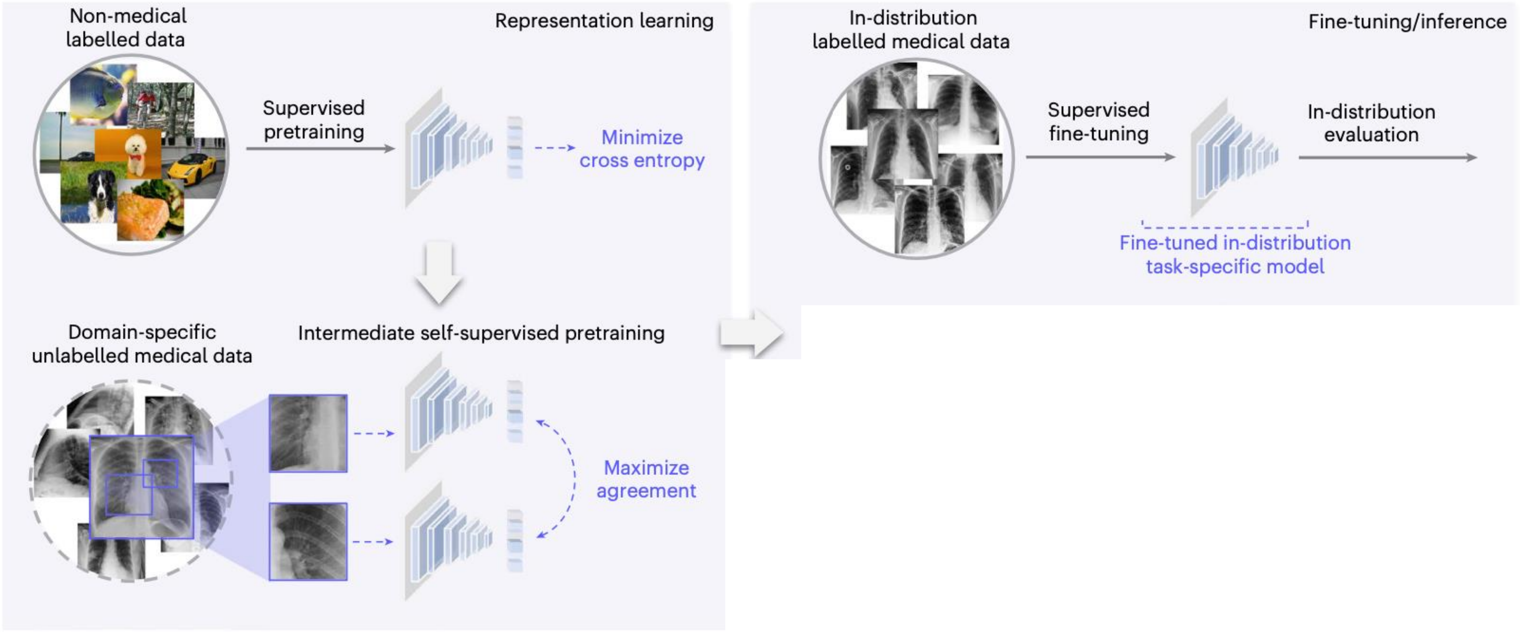
- REMEDIS, Robust and data-efficient generalization of self-supervised machine learning for diagnostic imaging, Azizi et al 2023, Nat. Biomed. Engineering

SUCCESSFUL APPLICATIONS OF SELF-SUPERVISED MODELS FOR HEALTHCARE



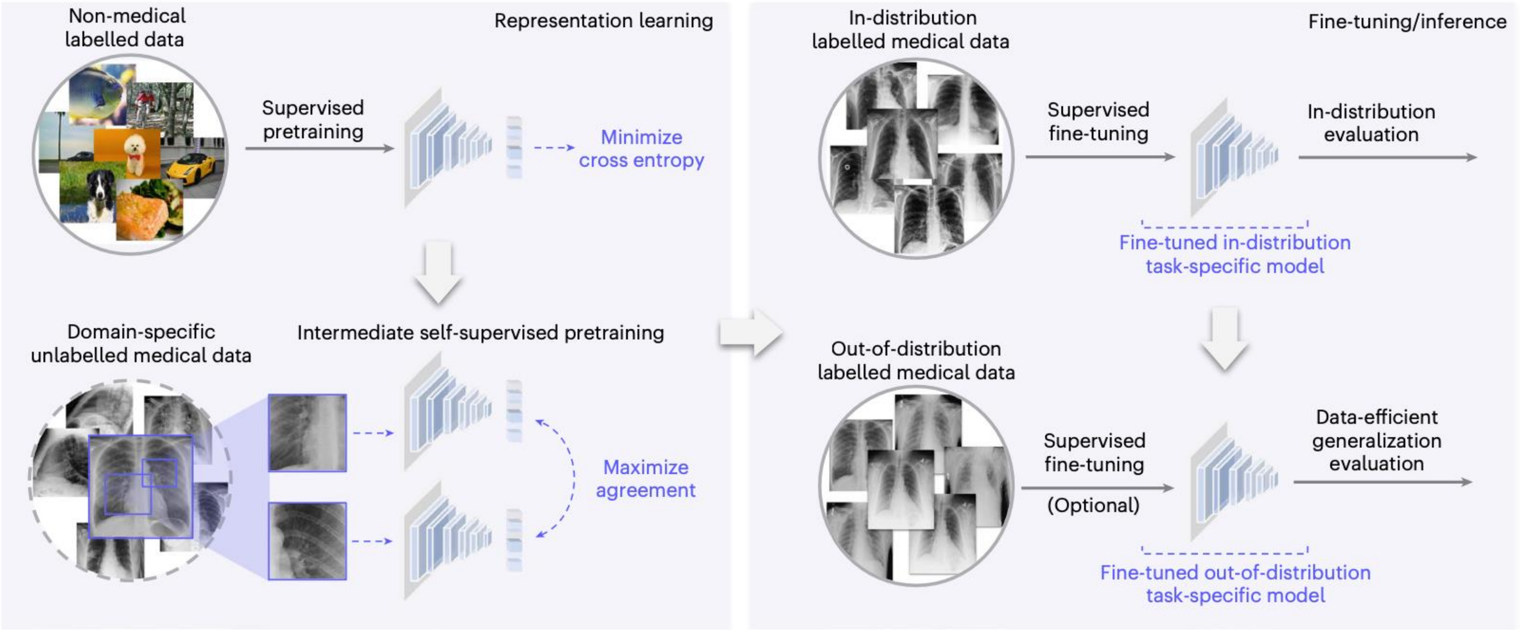
- REMEDIS, Robust and data-efficient generalization of self-supervised machine learning for diagnostic imaging, Azizi et al 2023, Nat. Biomed. Engineering

SUCCESSFUL APPLICATIONS OF SELF-SUPERVISED MODELS FOR HEALTHCARE



- REMEDIS, Robust and data-efficient generalization of self-supervised machine learning for diagnostic imaging, Azizi et al 2023, Nat. Biomed. Engineering

SUCCESSFUL APPLICATIONS OF SELF-SUPERVISED MODELS FOR HEALTHCARE



- REMEDIS, Robust and data-efficient generalization of self-supervised machine learning for diagnostic imaging, Azizi et al 2023, Nat. Biomed. Engineering

SUCCESSFUL APPLICATIONS OF SELF-SUPERVISED MODELS FOR HEALTHCARE

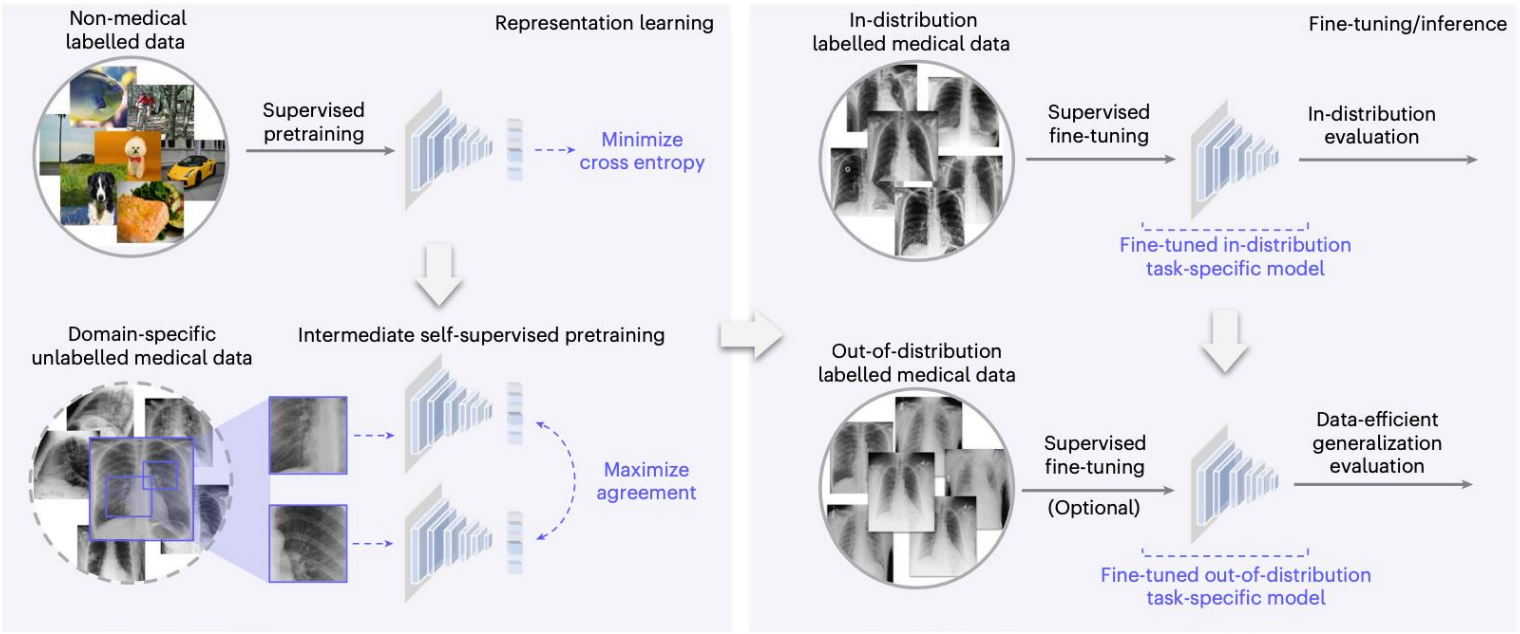


Fig. 1 | Overview of the REMEDIS approach for developing robust and efficient ML for medical imaging. REMEDIS starts with representations initialized using large-scale natural-image pretraining following the BiT method[52]. We then adapt the model to the medical domain using intermediate contrastive self-supervised learning without using any labelled medical data. Finally, we fine-tune the model to specific downstream medical-imaging ML tasks. We evaluate the ML model in both ID and OOD settings to establish the data-efficient generalization performance of the model.

The authors show that pretraining models with SSL:

→ improves performance overall

→ Improves model generalisation to external deployment settings

- REMEDIS, Robust and data-efficient generalization of self-supervised machine learning for diagnostic imaging, Azizi et al 2023, Nat. Biomed. Engineering

SUCCESSFUL APPLICATIONS OF SELF-SUPERVISED MODELS

A foundation model for generalizable disease detection from retinal images, Zhou et al., Nature 2023

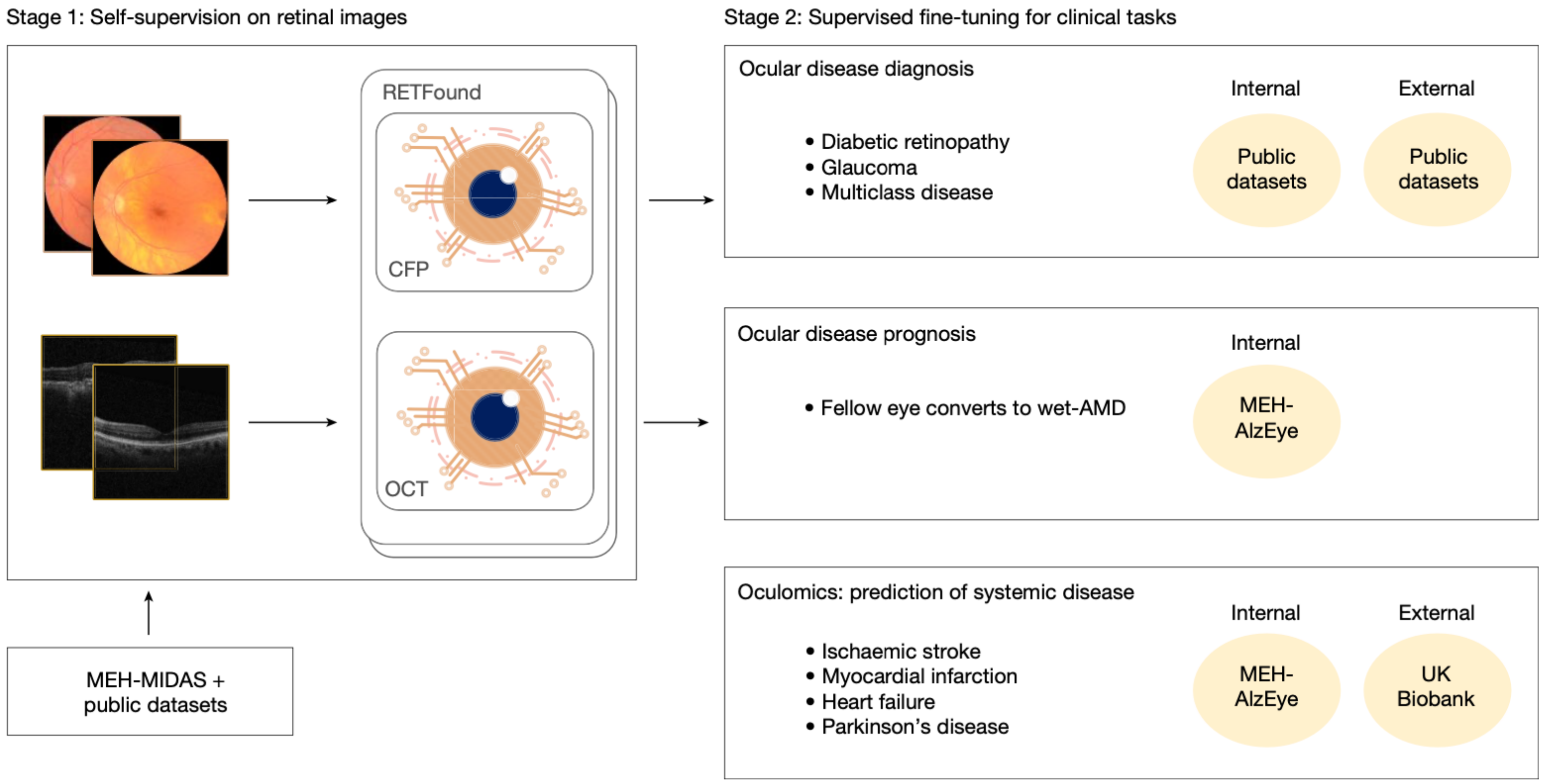


Fig. 1 | Schematic of development and evaluation of the foundation models (RETFound). Stage one constructs RETFound by means of SSL, using CFP and OCT from MEH-MIDAS and public datasets. Stage two adapts RETFound to downstream tasks by means of supervised learning for internal and external evaluation.

RESOURCES AND REFERENCES

- Summary of contrastive learning methods
  - Blogs
    - https://encord.com/blog/guide-to-contrastive-learning/#:~:text=Contrastive%20learning%20is%20an%20approach,instances%20should%20be%20farther%20apart.
    - https://lilianweng.github.io/posts/2021-05-31-contrastive/
- SimCLR
  - Blogs
    - https://amitness.com/2020/03/illustrated-simclr/
  - Paper: http://proceedings.mlr.press/v119/chen20j/chen20j.pdf
- BYOL
  - Paper https://papers.nips.cc/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf
- DINO
  - Blog: https://ai.meta.com/blog/dino-paws-computer-vision-with-self-supervised-transformers-and-10x-more-efficient-training/
  - Paper: https://arxiv.org/pdf/2104.14294.pdf
  - Code: https://github.com/facebookresearch/dino

- MAE
  - Blog / video
    - https://itnext.io/masked-autoencoders-are-scalable-vision-learners-122a75b54470
    - https://www.youtube.com/watch?v=Dp6iICL2dV
  - Paper: https://openaccess.thecvf.com/content/CVPR2022/papers/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.pdf
- i-Jepa
  - Blog: https://ai.meta.com/blog/yann-lecun-ai-model-i-jepa/
  - Paper: https://arxiv.org/abs/2301.08243
- Medical applications
  - REMEDIS:
    - Blog: https://blog.research.google/2023/04/robust-and-efficient-medical-imaging.html
    - Paper: https://www.nature.com/articles/s41551-023-01049-7
  - RetFound: https://www.nature.com/articles/s41586-023-06555-x