CS-EJ3311 –

Deep Learning with Python

Regularization

Alexander Jung

FITech NETWORK UNIVERSITY

What I want to teach you today:

- basic idea of regularization
- regularization via data augmentation
- regularization via transfer learning

What is ML ?

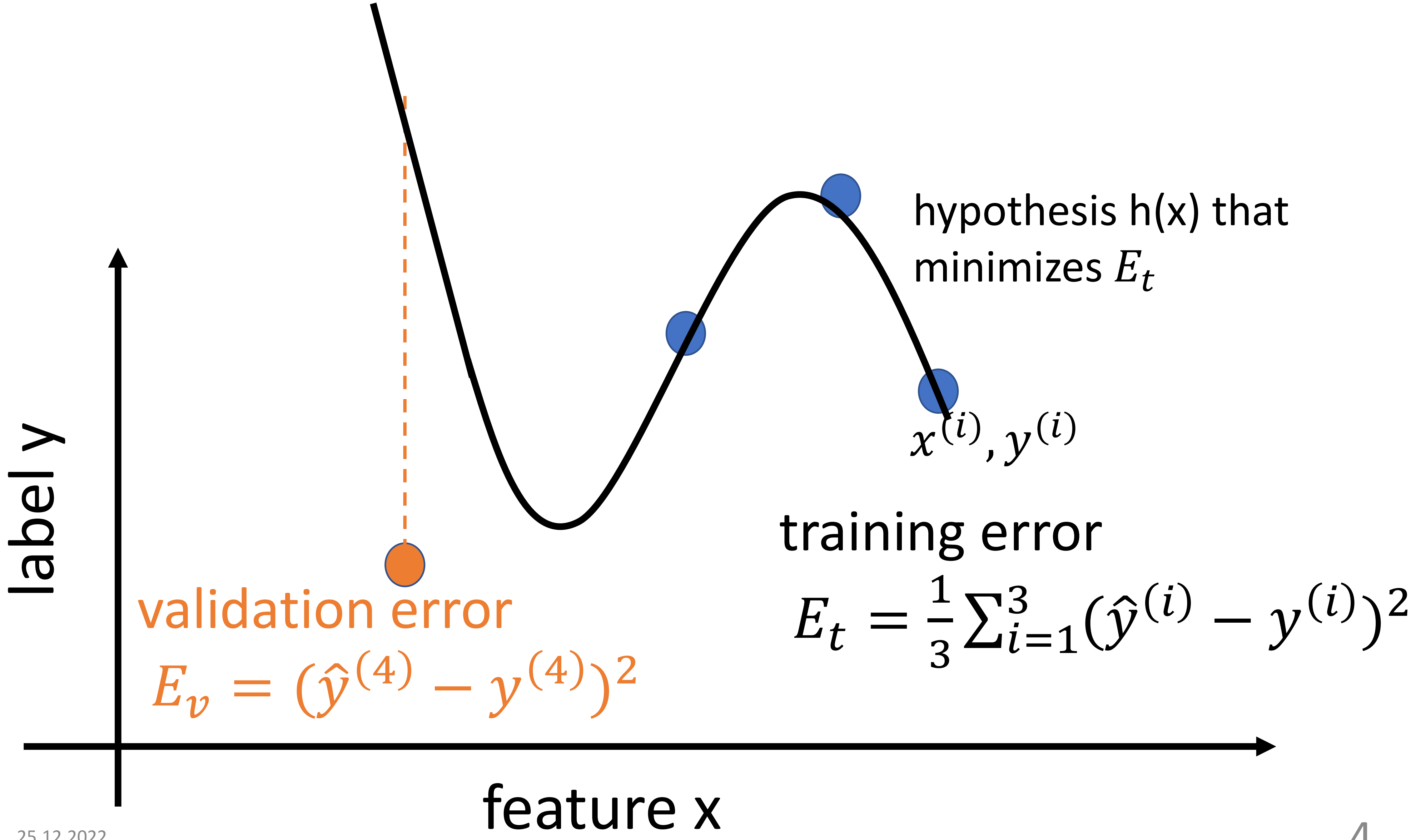
informal: learn hypothesis out of a hypothesis space or "model" that incurs minimum loss when predicting labels of datapoints based on their features

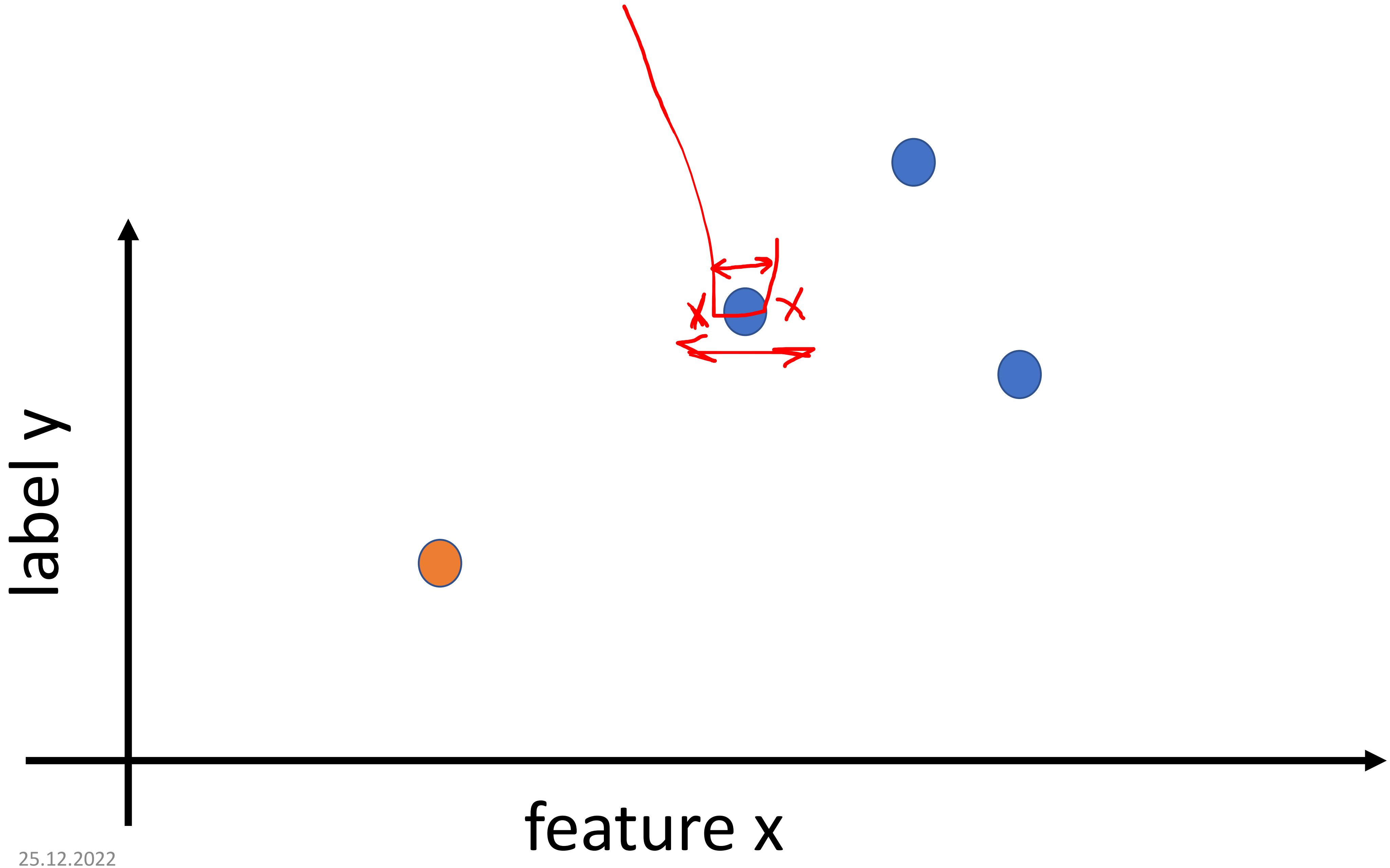
"training error"

$$\hat{h} = \operatorname*{argmin}_{h \in \mathcal{H}} \mathcal{E}(h|\mathcal{D})$$

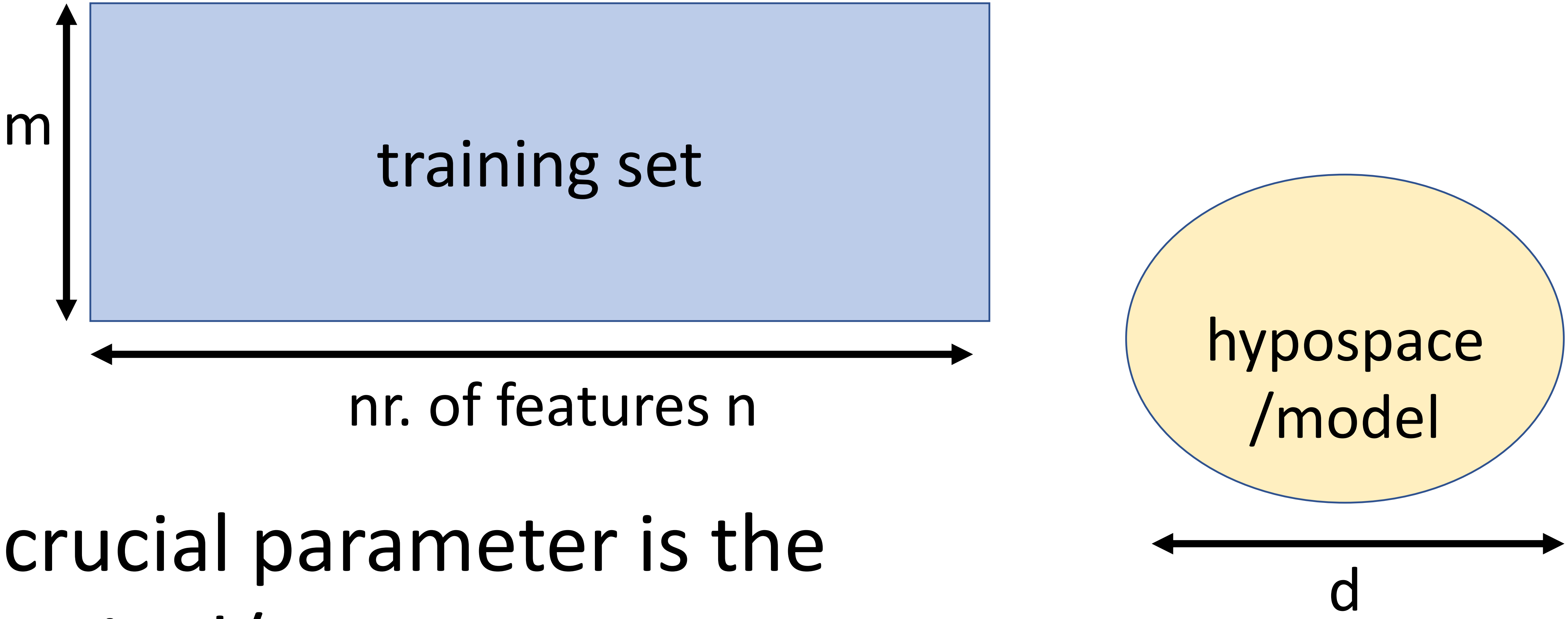
$$\overset{(\boxed{2.12})}{=} \operatorname*{argmin}_{h \in \mathcal{H}} (1/m) \sum_{i=1}^{m} \mathcal{L}((\mathbf{x}^{(i)}, y^{(i)}), h).$$

see Ch. 4.1 of mlbook.cs.aalto.fi

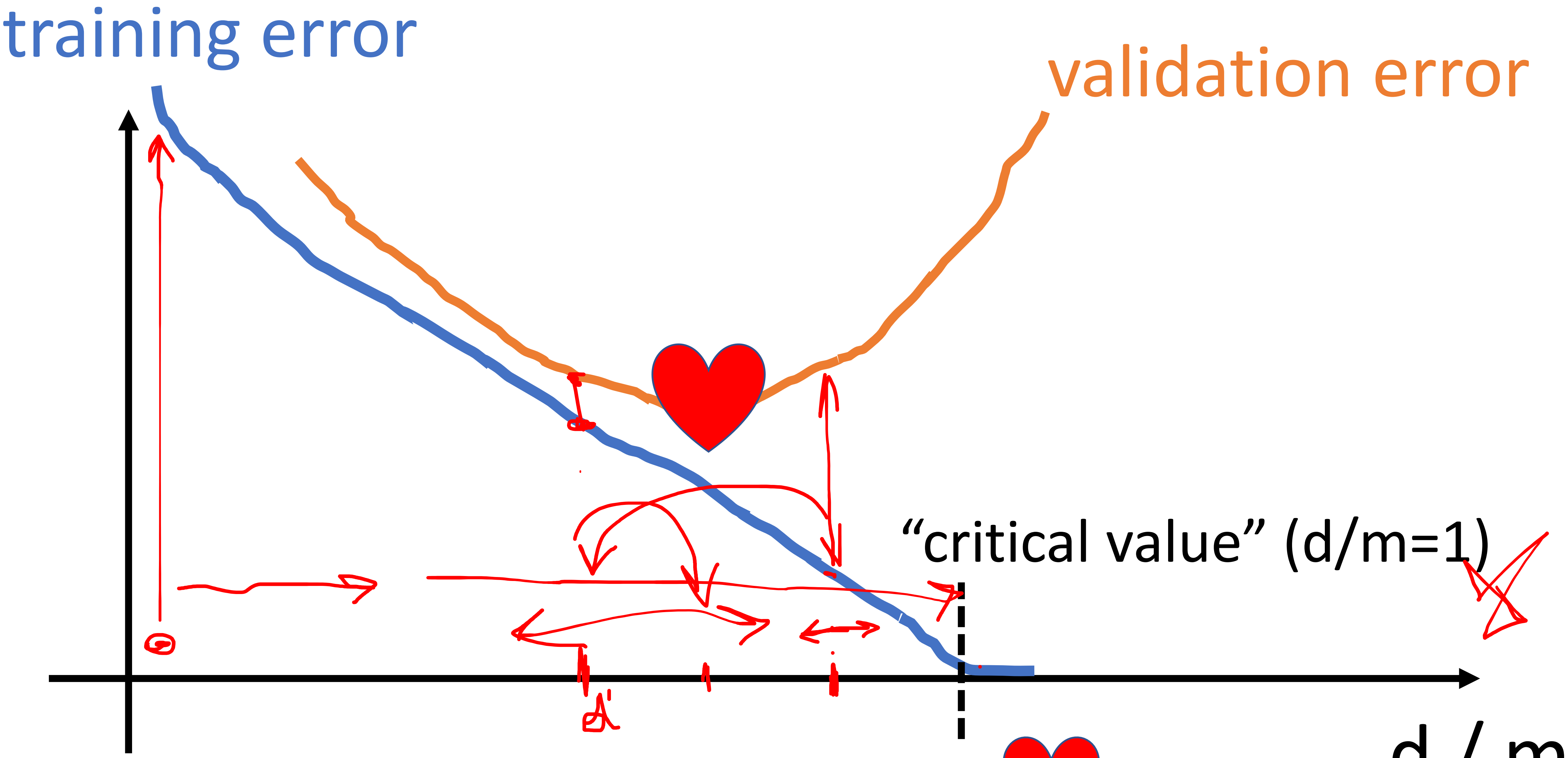

hypothesis h(x) that
minimizes E_t
$x^{(i)}, y^{(i)}$
training error
$E_t = \frac{1}{3}\sum_{i=1}^{3}(\hat{y}^{(i)} - y^{(i)})^2$
validation error
$E_v = (\hat{y}^{(4)} - y^{(4)})^2$
label y
feature x


label y
feature x

Data and Model Size



crucial parameter is the ratio d/m



adjust model and/or data to reach ♥

model data

bring d/m below critical value 1:

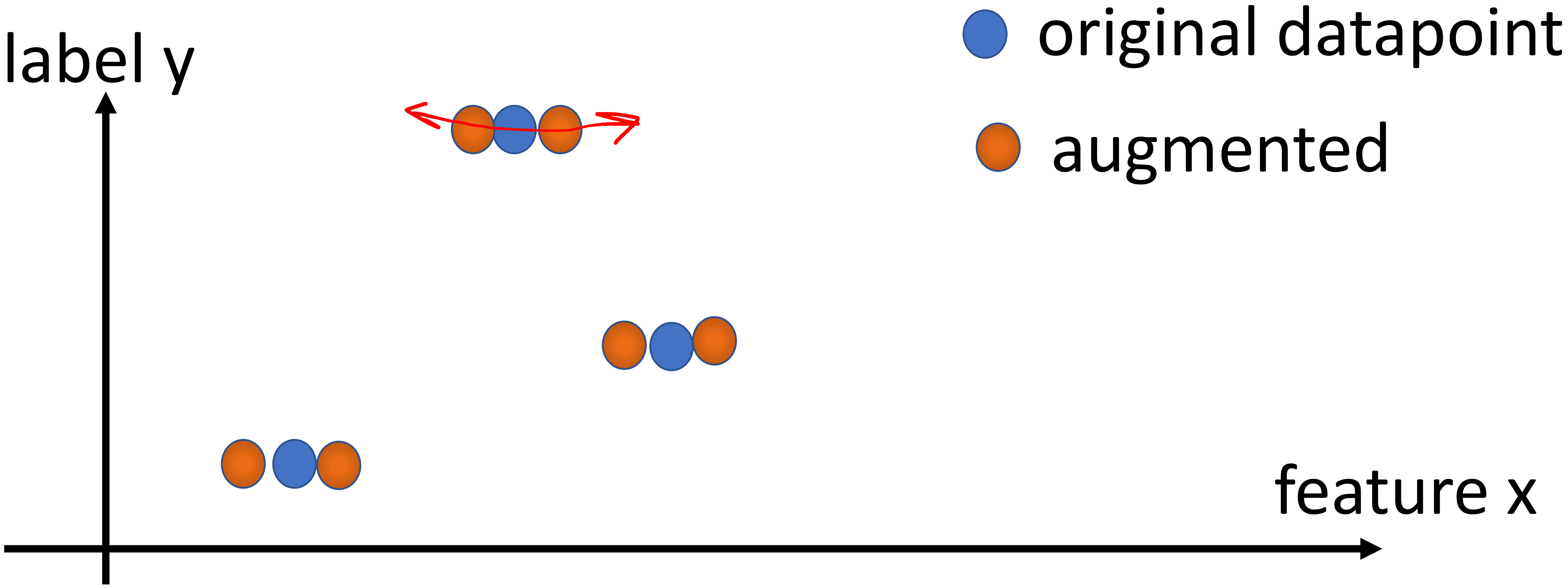
- increase m by using more training data
- decrease d by using smaller hypothesis space

Data Augmentation

bring d/m below critical value 1:

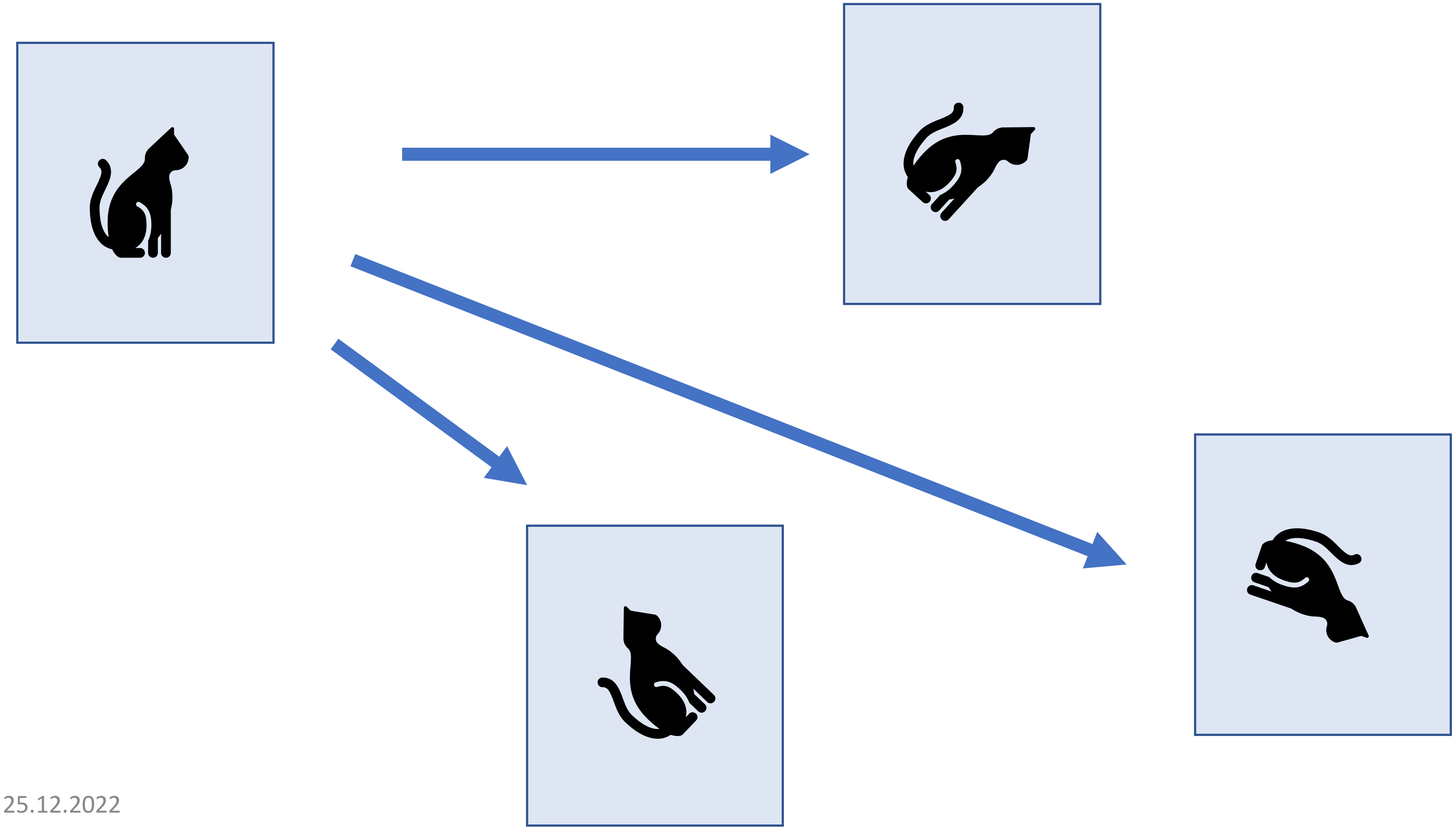
- increase m by using more training data
- decrease d by using smaller hypothesis space

add a bit of noise to features

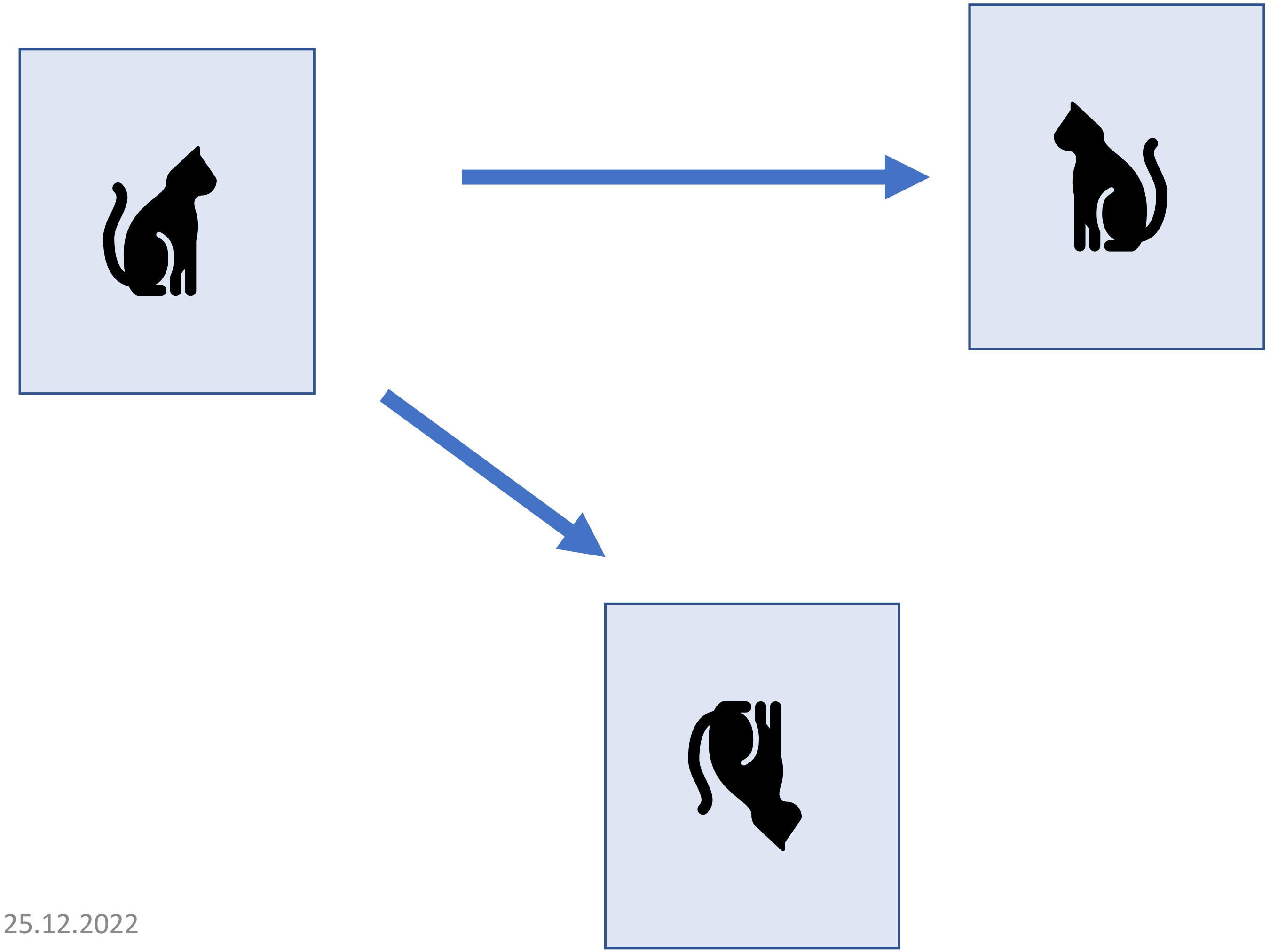


we have enlarged dataset by factor 3 !

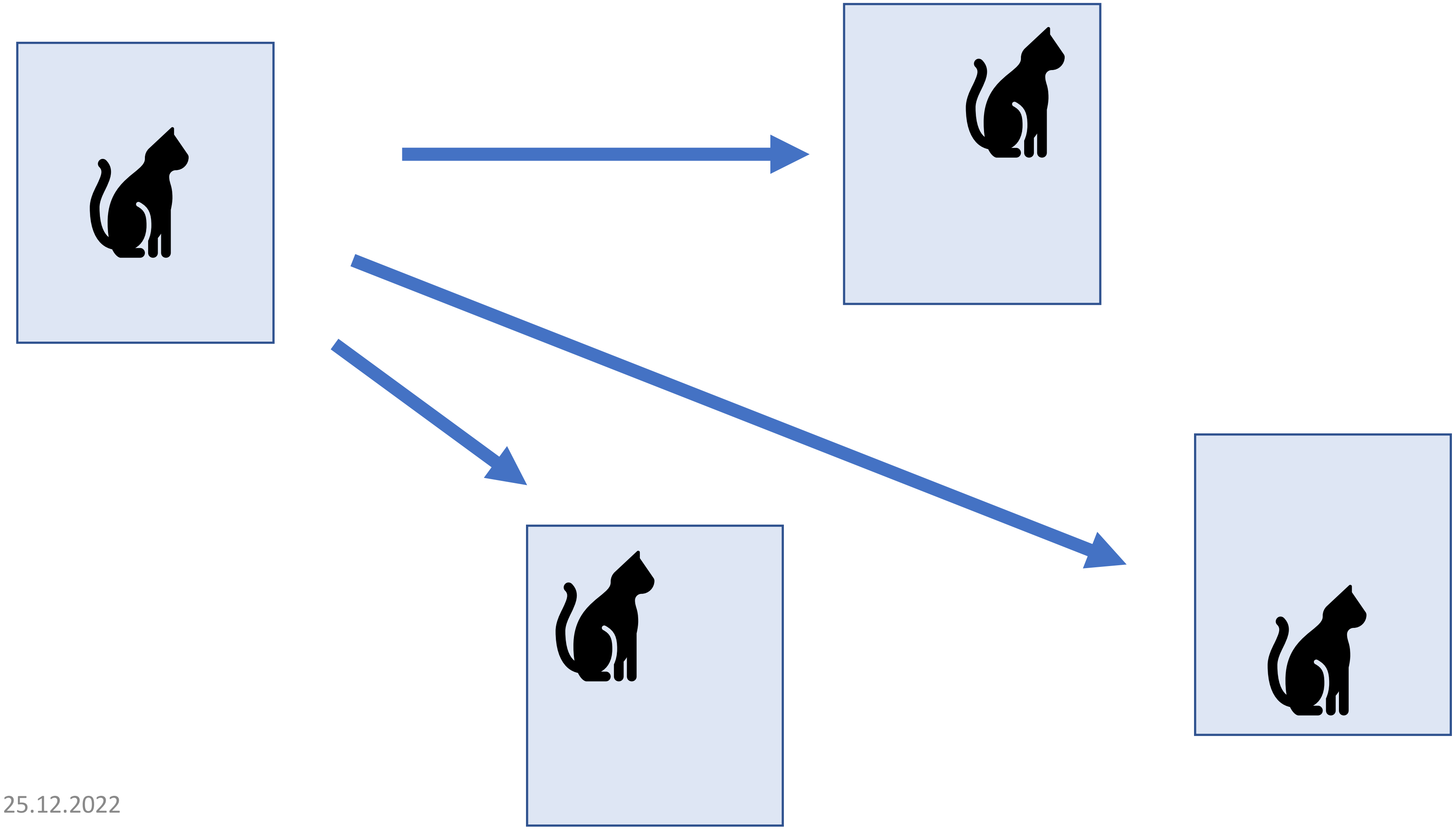
rotated cat image is still cat image



flipped cat image is still cat image



shifted cat image is still cat image



In [19]:
```
plt.figure(figsize=(8, 8))
for images, _ in train_ds.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[0].numpy())
        plt.axis("off")
plt.show()
```

Transfer Learning

bring d/m below critical value 1:

- increase m by using more training data
- decrease d by using smaller hypothesis space

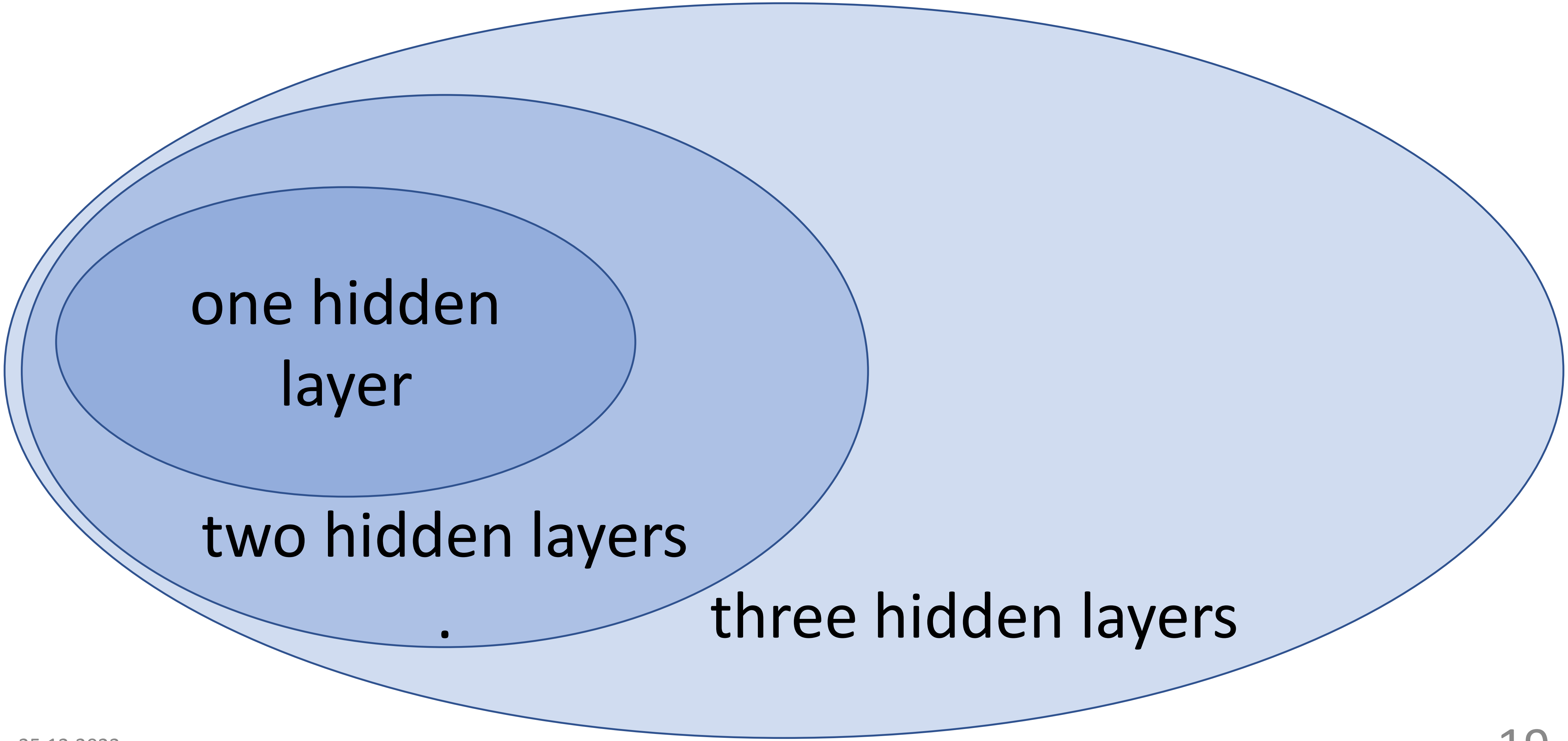
replace original ERM

$$\min_{h \epsilon \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\left(x^{(i)}, y^{(i)}\right), h\right)$$

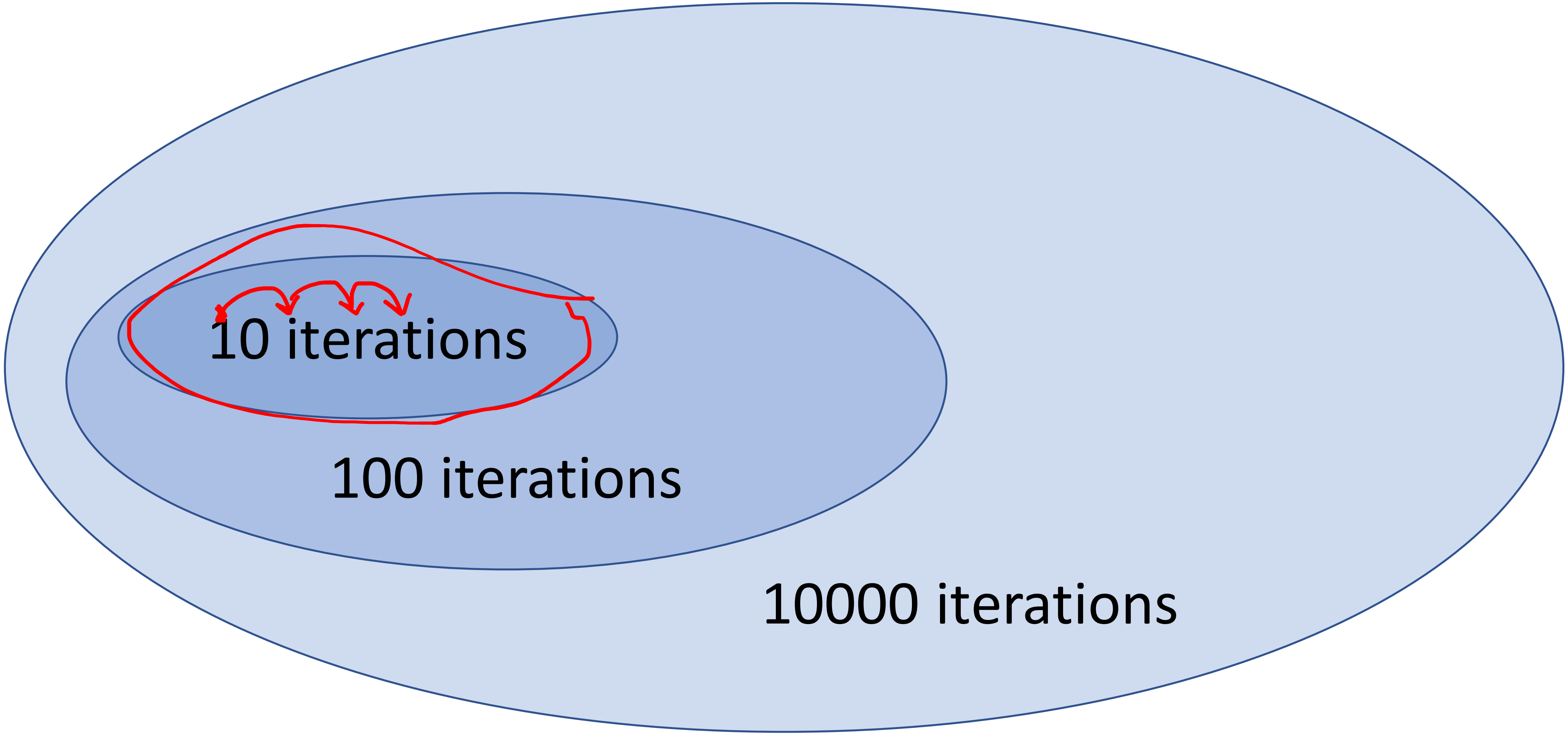
with ERM on smaller $\widehat{\mathcal{H}} \subset \mathcal{H}$

$$\min_{h \epsilon \widehat{\mathcal{H}}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\left(x^{(i)}, y^{(i)}\right), h\right)$$

Prune Network Architecture

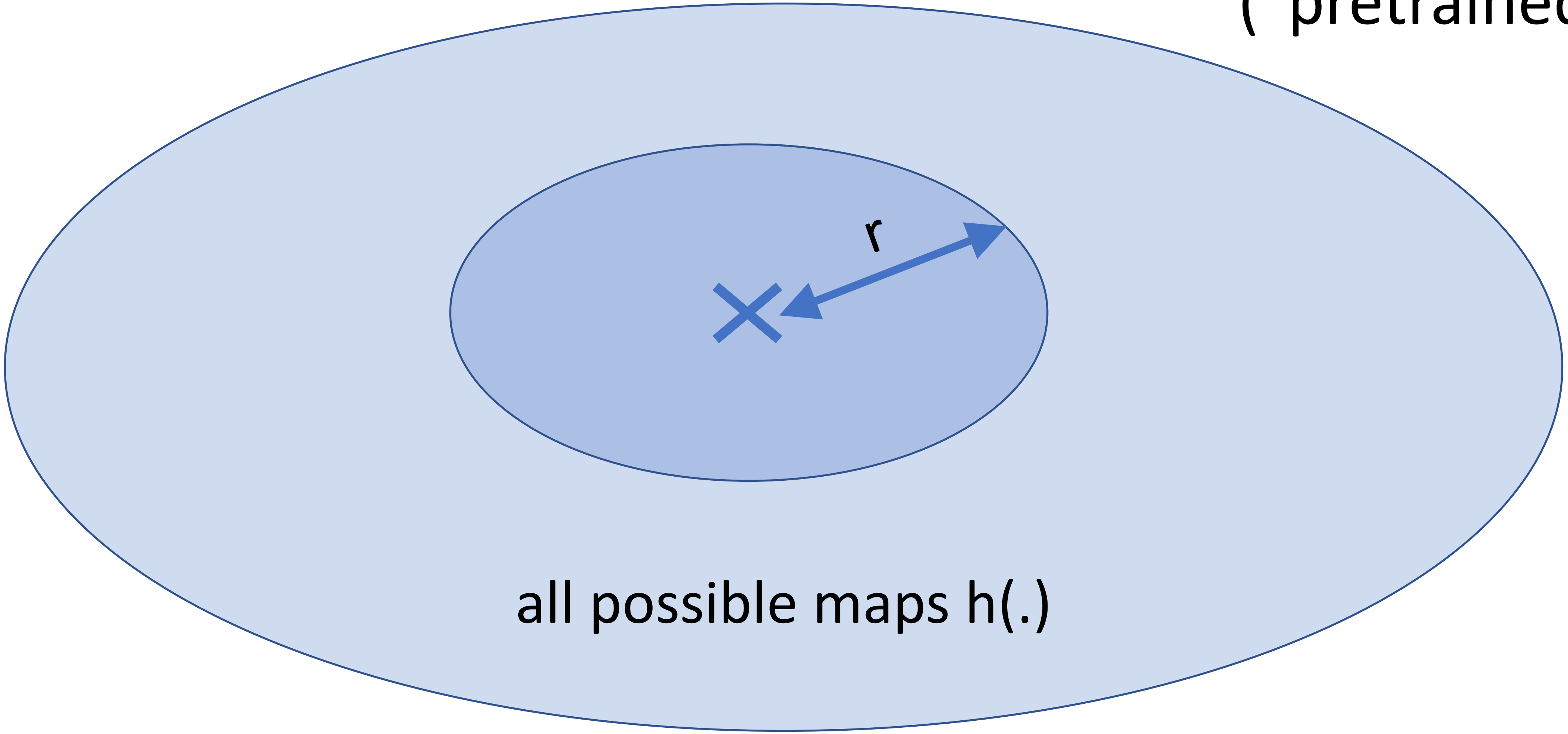


Prune Hypospace by Early Stopping

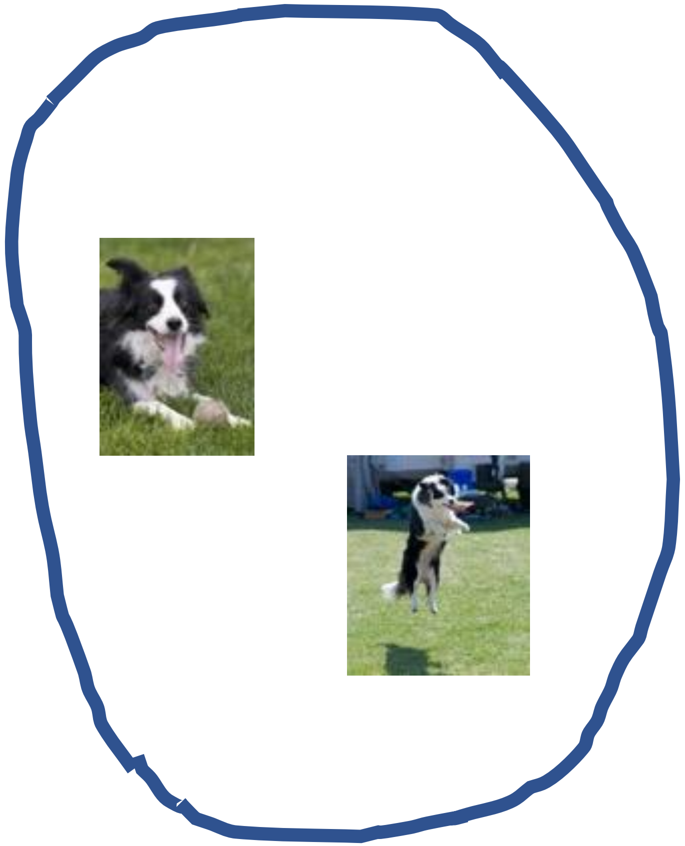


Transfer Learning

reference hypothesis $\hat{h}$ ("pretrained net")

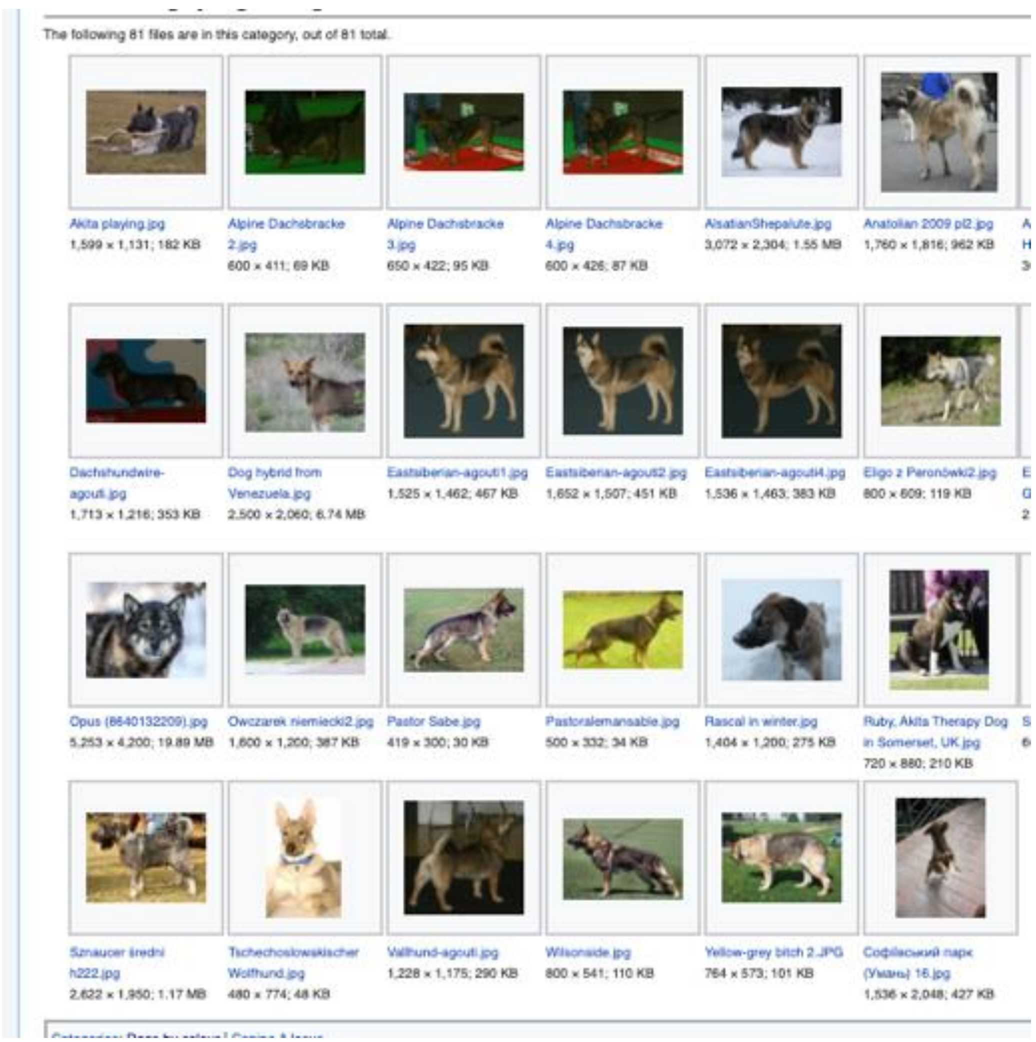


- Problem I: classify image as “shows border collie” vs. “not”
- Problem II: classify image as “shows a dog” vs. “not”
- ML Problem I is our main interest
- only little training data $\mathcal{D}^{(1)}$ for Problem I
- much more labeled data $\mathcal{D}^{(2)}$ for Problem II
- pre-train a hypothesis on $\mathcal{D}^{(2)}$, fine-tune on $\mathcal{D}^{(1)}$



$\mathcal{D}^{(1)}$

learn h by fine-tuning $\hat{h}$



$\mathcal{D}^{(2)}$

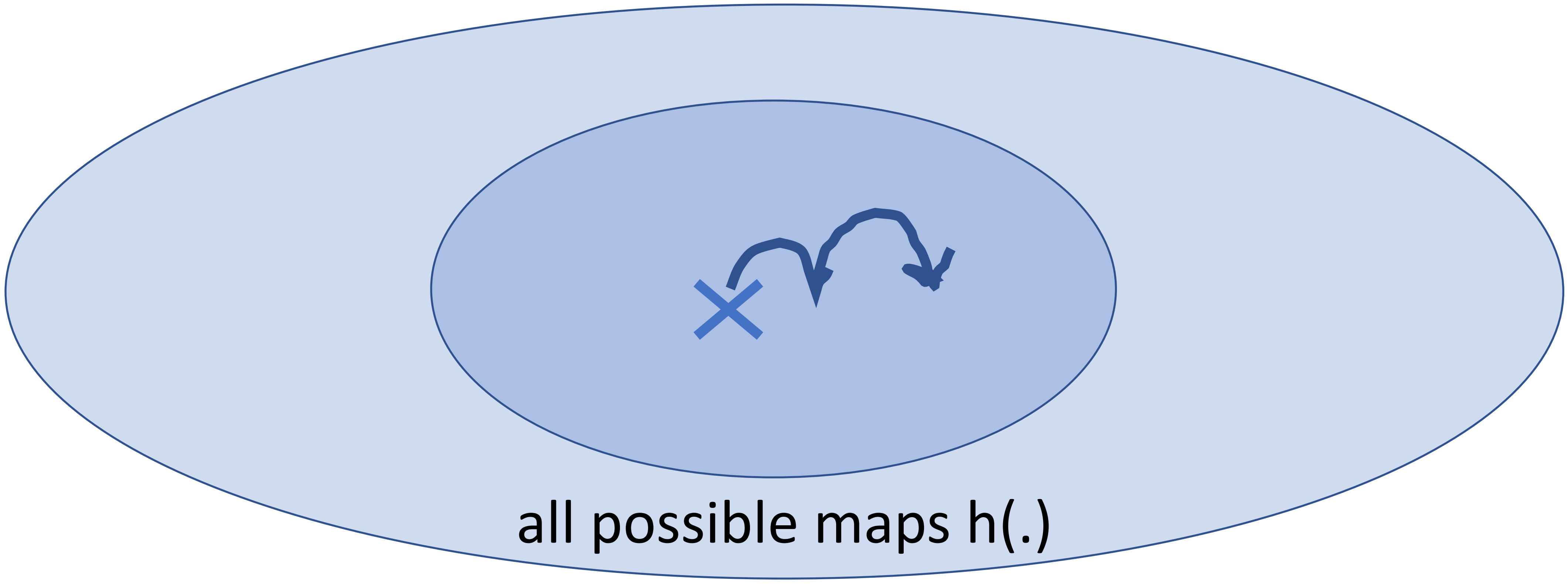
pre-train hypothesis $\hat{h}$

$$\underbrace{\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\left(x^{(i)}, y^{(i)}\right), h\right)}_{\text{fine tuning on } \mathcal{D}^{(1)}} + \underbrace{\lambda d(h, \hat{h})}_{\text{distance to hypothesis } \hat{h} \text{ which is pre-trained on } \mathcal{D}^{(2)}}$$

fine tuning on $\mathcal{D}^{(1)}$

distance to hypothesis $\hat{h}$ which is pre-trained on $\mathcal{D}^{(2)}$

Fine Tuning a Pretrained Net



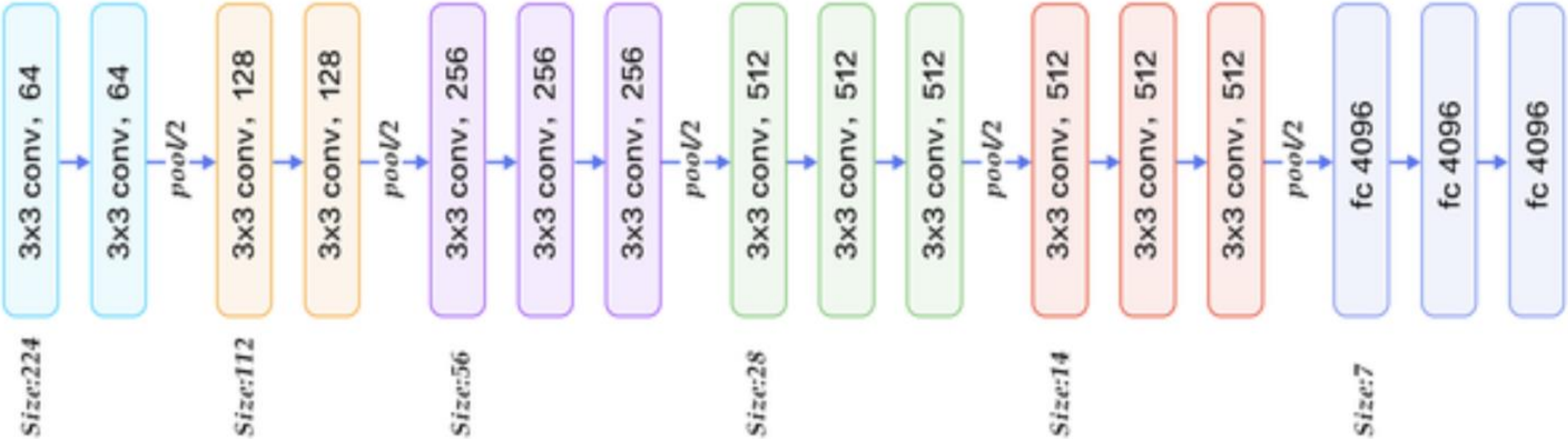
learning rate/step size used during fine tuning determines effective model size

```python
tf.keras.applications.vgg16.VGG16(
    include_top=True, weights='imagenet', input_tensor=None,
    input_shape=None, pooling=None, classes=1000,
    classifier_activation='softmax'
)
```

https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16/VGG16

Layer-Wise Fine Tuning

fine –tune deeper layers
“freeze” input layers



“cat”

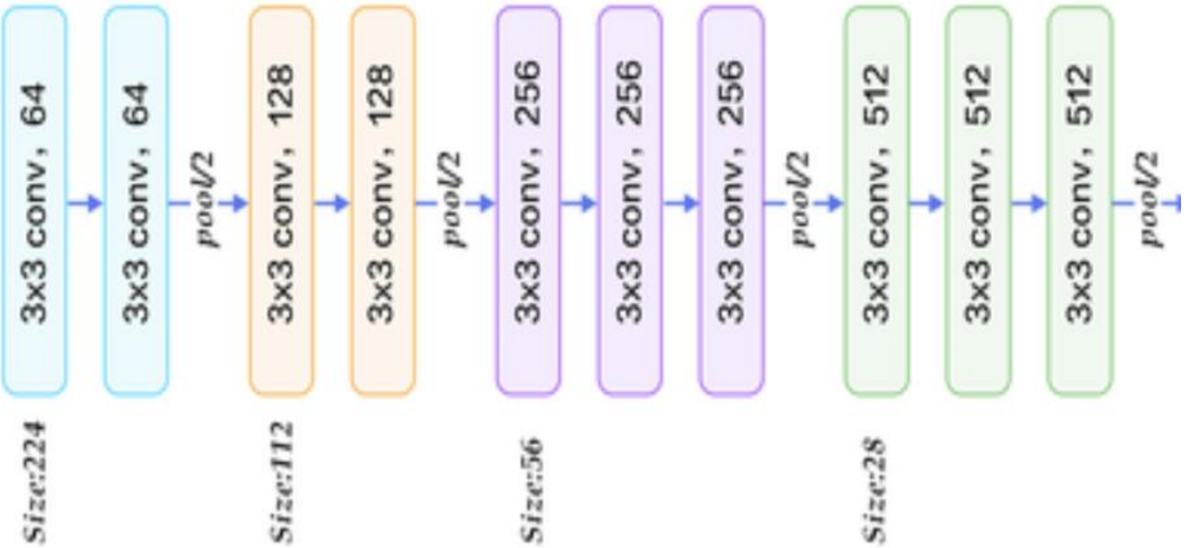
https://www.quora.com/What-is-the-VGG-neural-network

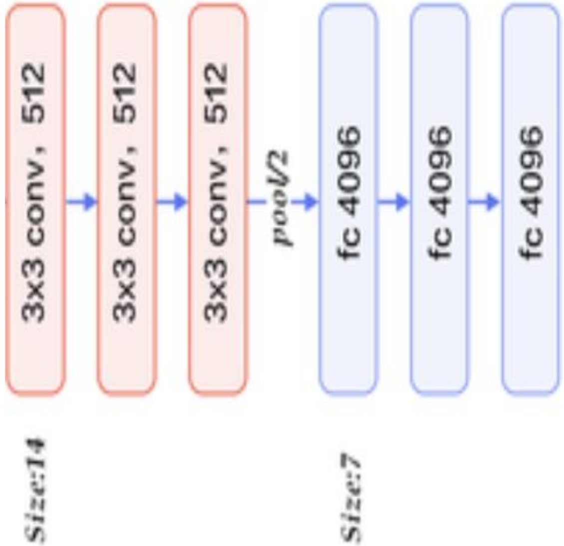
Feature Extraction

“frozen” input layers perform feature extraction

“feature extractor” or ”base” model

“head”





“cat”

https://www.quora.com/What-is-the-VGG-neural-network

```
base_model = keras.applications.Xception(
    weights='imagenet',  # Load weights pre-trained on ImageNet.
    input_shape=(150, 150, 3),
    include_top=False)  # Do not include the ImageNet classifier at the top.
```

Then, freeze the base model.

```
base_model.trainable = False
```

https://keras.io/guides/transfer_learning/

Questions ?