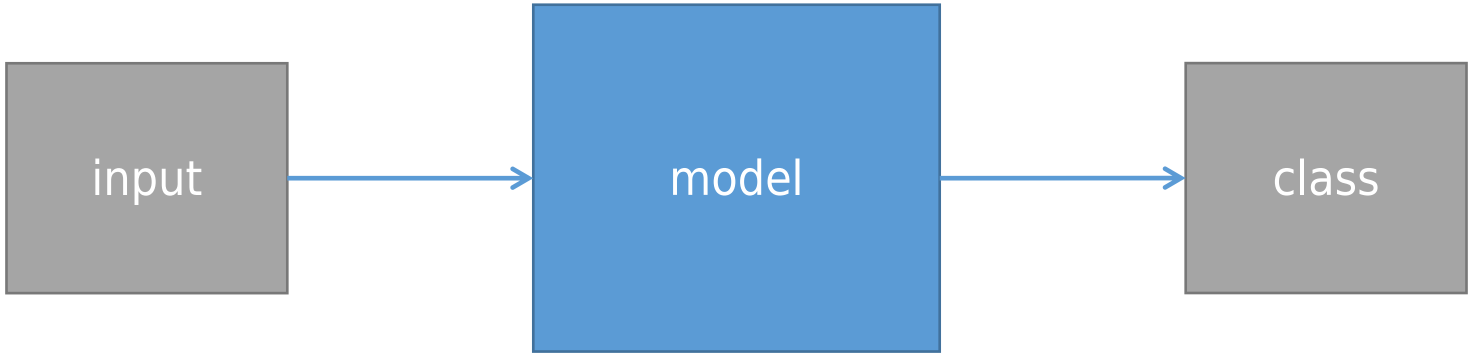
CrossEntropy Loss

How to train a classifier

Classifier

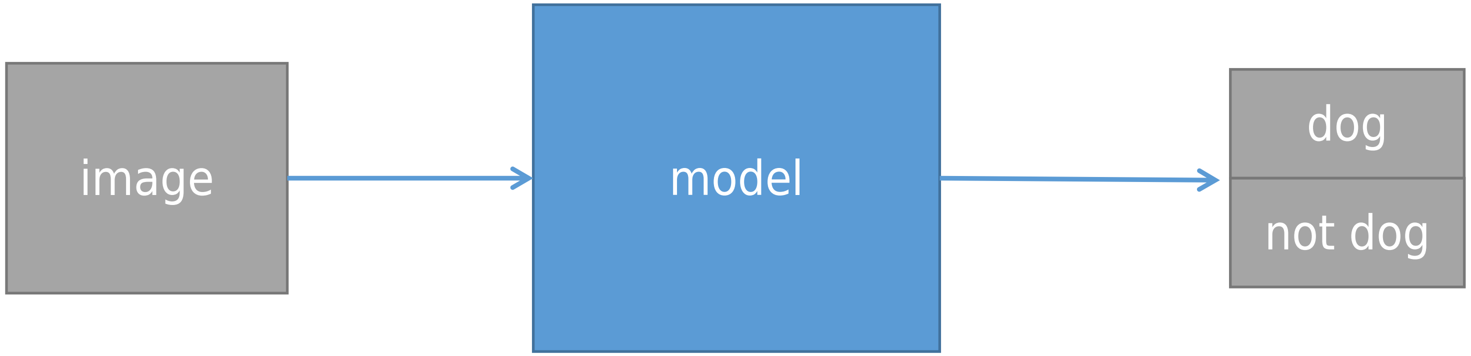
A **classifier** is a type of **model** that is **trained** to **categorize** input data into **classes**.

The **most used loss** function to **train** a **classifier** is the **crossentropy** loss.



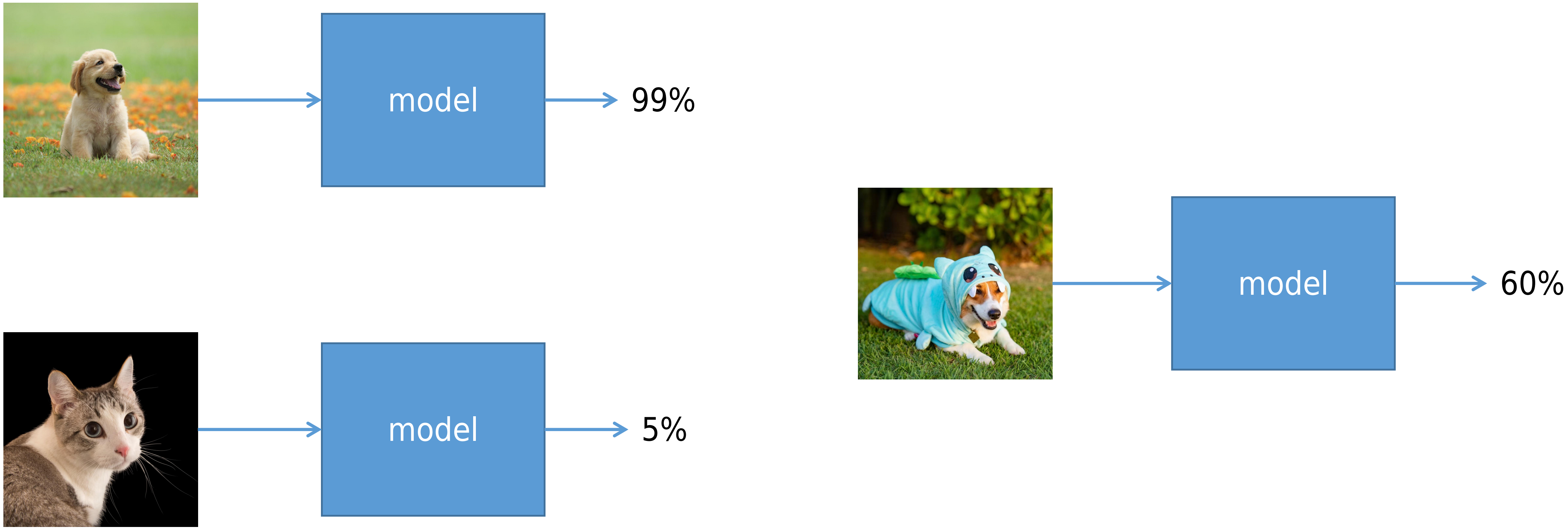
Binary Classifier

A **binary classifier** is a **classifier** which can only **distinguish** between **two classes**.



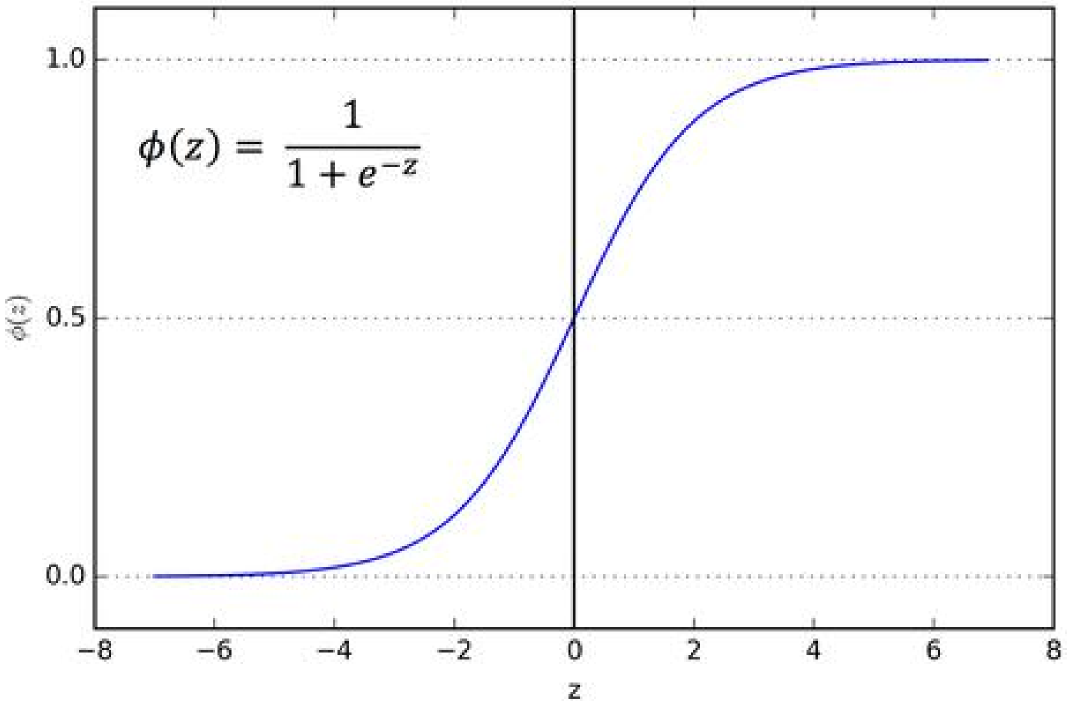
Binary Classifier: Logistic Regression

In the **logistic regression** scenario, we aim to **train** our **network** to **output** the **probability** of the **input being** in the **target class**.



Logistic Regression: Sigmoid Function

The **sigmoid function**, also **known** as the **logistic function**, is a function that **maps** any **real-valued number to** a value between **0** and **1**. It is commonly **used** in **logistic regression** to model the **probability** that a given **input belongs** to a particular **class**



Binary Classifier: Logistic Regression loss

Given a **network output** $o = model(x)$ and a **target class** t (which can be 0 or 1).
We can **transform** o **into a probability** using the **sigmoid** function $\sigma(o)$.
The **probability** of x of **being** of **class** t is then $\sigma(o)$ and the **probability** of **not being** of **class** t is $1 - \sigma(o)$.

The Logistic Regression loss function is:

$$loss = -(t \bullet log(\sigma(o)) + (1-t) \bullet log(1-\sigma(o)))$$

Why this expression? why the minus sign? and why log?

Binary Classifier: Logistic Regression loss

$$loss = -(t \bullet log(\sigma(o)) + (1-t) \bullet log(1-\sigma(o)))$$

If target class is 0 and the predicted probability $\sigma(o)$ is 0.2

$$loss = -(0 \bullet log(0.2) + 1 \bullet log(1-0.2)) = -log(0.8)$$

If target class is 0 and the predicted probability $\sigma(o)$ is 0.8

$$loss = -(0 \bullet log(0.8) + 1 \bullet log(1-0.8)) = -log(0.2)$$

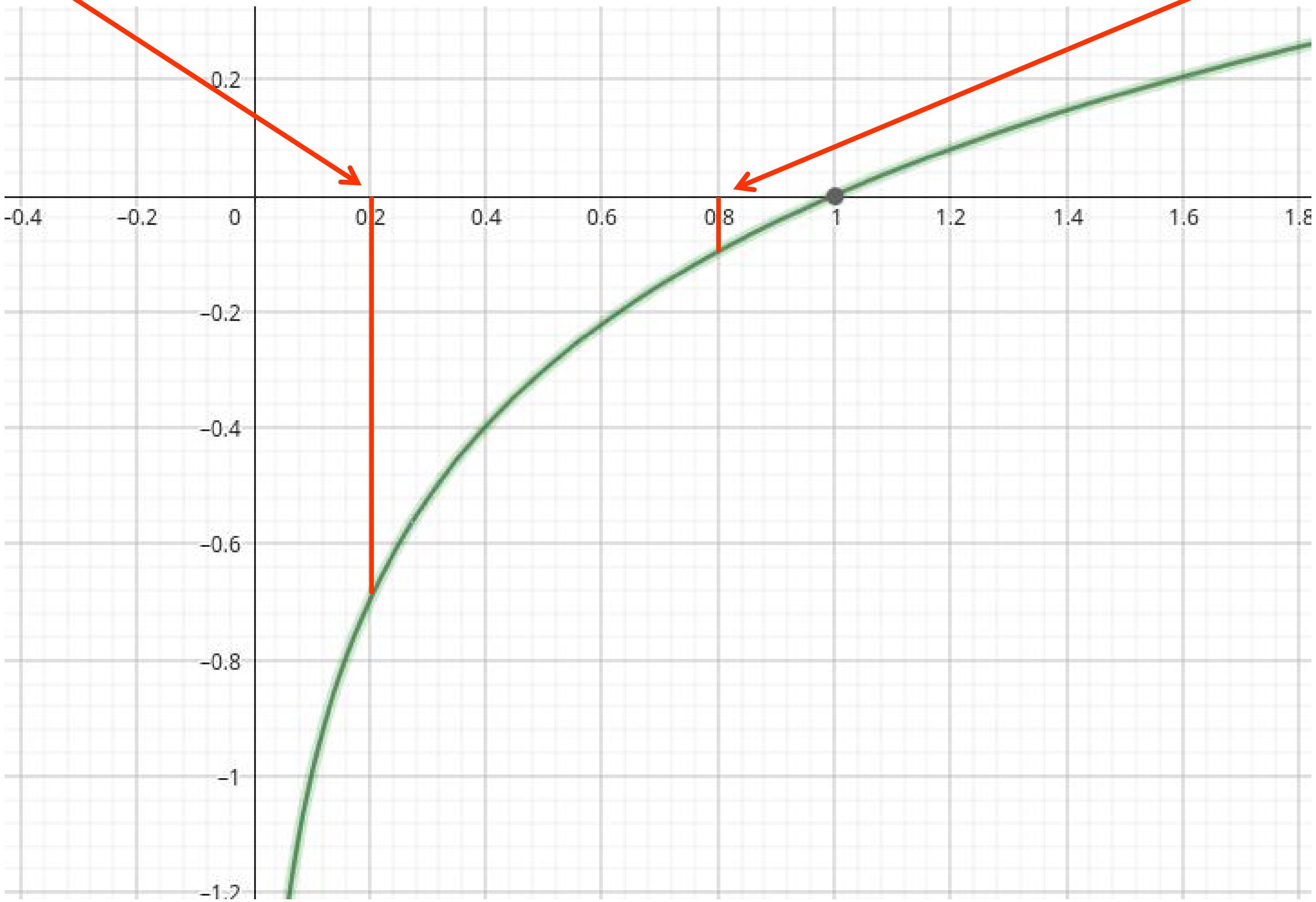
Binary Classifier: Logistic Regression loss

target 0 and predicted 0.8 = -log(0.2)

big error

target 0 and predicted 0.2 = -log(0.8)

small error



The error is exponentially larger w.r.t. the predicted probability error

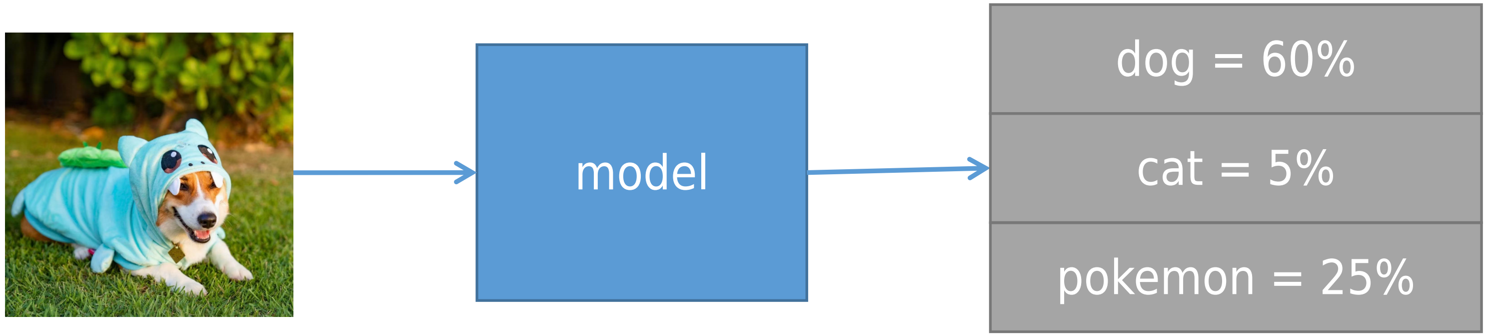
More than two classes

A **classfier** is **trained** to **classify** an **input** into **more** than two **classes**.

The **output** of the model **should** be a **categorial distribution**.

A **categorical distribution** is a probability distribution used to model the **likelihood** of **distinct classes**, **each class** has an associated **probability**.

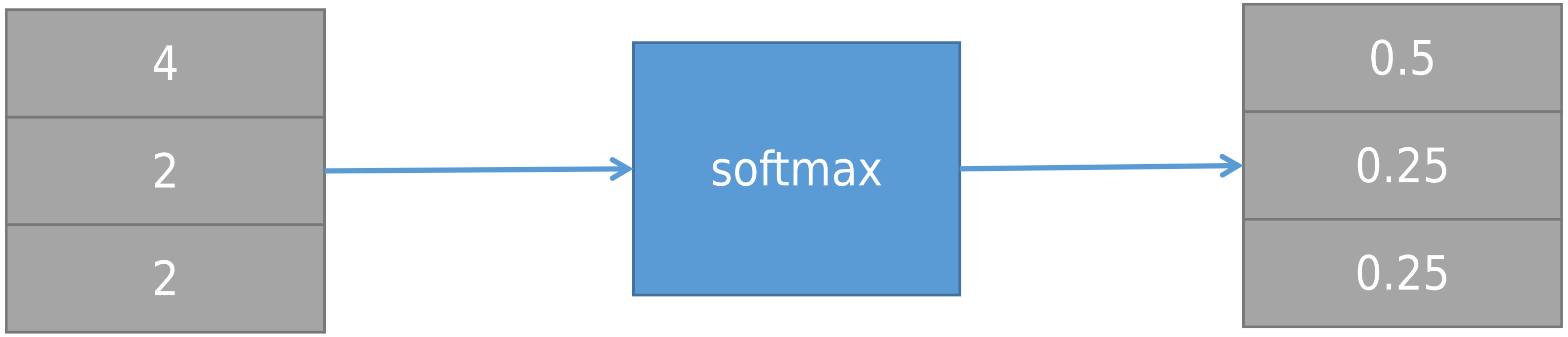
The **probabilities sum** to **1**, reflecting the **exclusive** nature of the **categories.**



Classifier: the softmax function

The **softmax function** is a function that takes as **input** a **vector** of real **numbers** and transforms it **into** a **categorical probability** distribution.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$



Classifier: the crossentropy loss

Generalization over **multiple classes** of the **logistic regression** loss:

$$loss = -\sum_{c=0}^{n} t_c \bullet log(o_c)$$

where t is the one-hot encoded target class.

target = 2 $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

target = 0 $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Classifier: the crossentropy loss

$$loss = -\sum_{c=0}^{n} t_c \bullet log(o_c)$$

Problem with 4 classes, target class is 1, output distribution is [0.1, 0.2, 0.2, 0.5]

$$t = [0\ 1\ 0\ 0]$$

$$loss = -log(0.2)$$

one-hot encoding
of class 1

Crossentropy loss in PyTorch

The **crossentropy** loss is implemented in **PyTorch** in the class **torch.nn.CrossEntropyLoss**.

The forward function of the **CrossEntropyLoss** module accepts a **tensor** in the **shape [batch, classes] and** a **tensor** of classes in the **shape [batch]**

```
import torch

loss_fn = torch.nn.CrossEntropyLoss()

logits = torch.randn(8, 5)
classes = torch.tensor([2, 1, 4, 2, 0, 3, 2, 0])

print(loss_fn(logits, classes))
```

```
import torch

loss_fn = torch.nn.NLLLoss()

logits = torch.randn(8, 5)
probs = torch.softmax(logits, dim=1)
log_probs = torch.log(probs)

classes = torch.tensor([2, 1, 4, 2, 0, 3, 2, 0])

print(loss_fn(log_probs, classes))
```