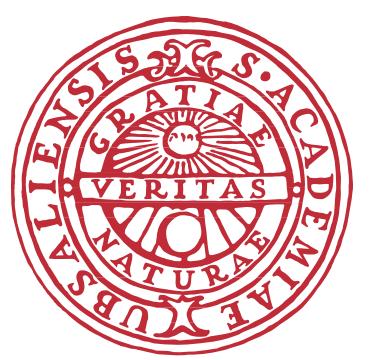
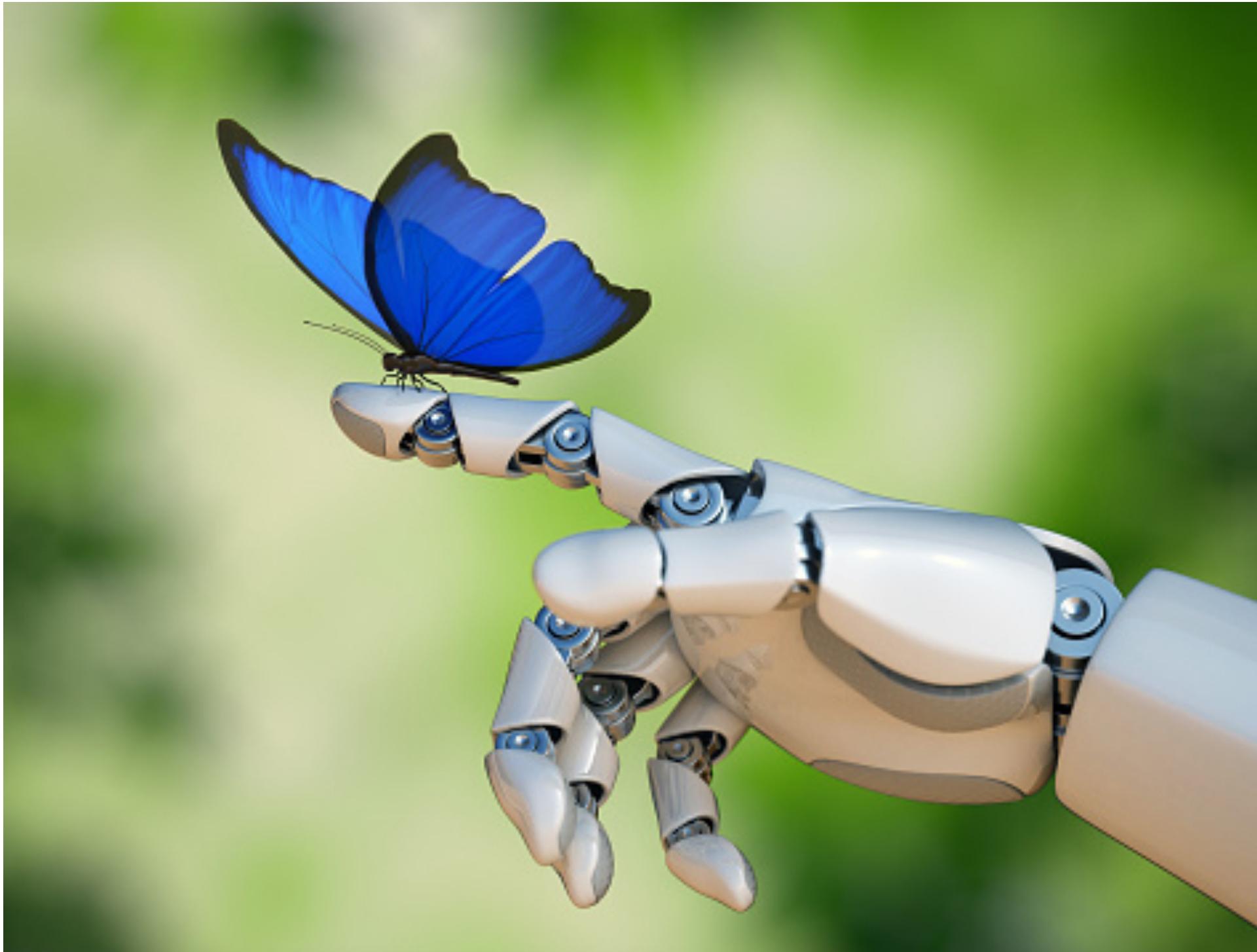


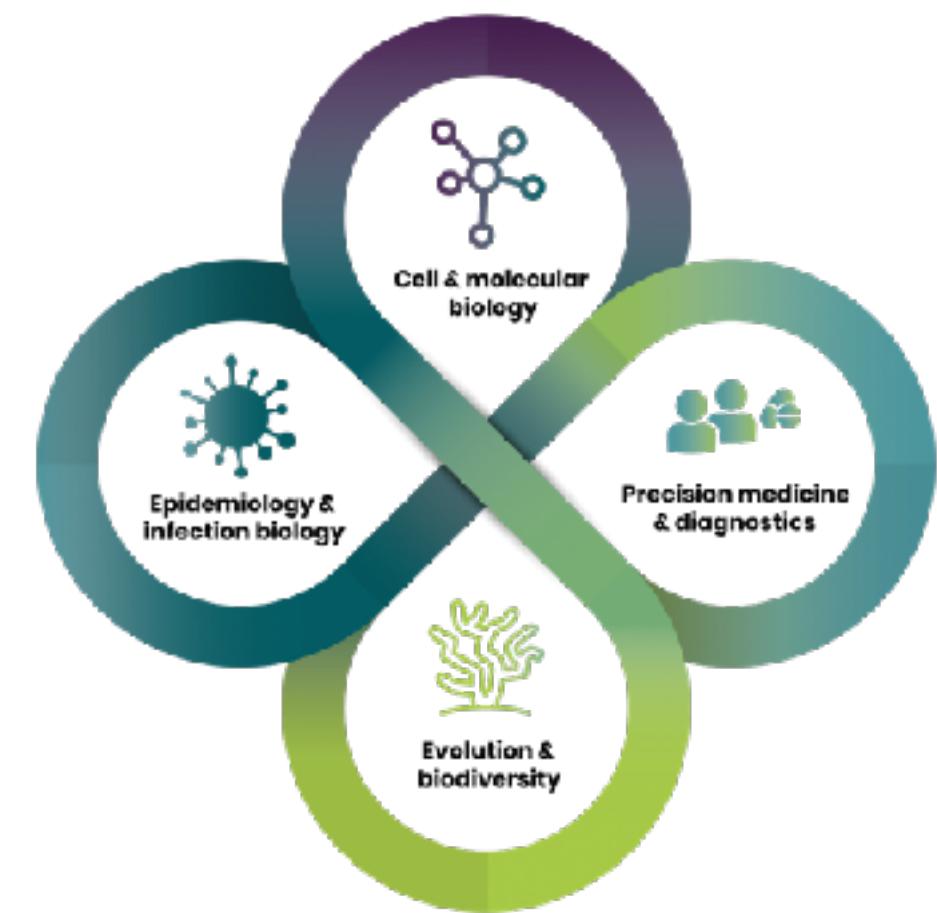
Neural Networks for biodiversity research



UPPSALA
UNIVERSITET



SciLifeLab



Overview

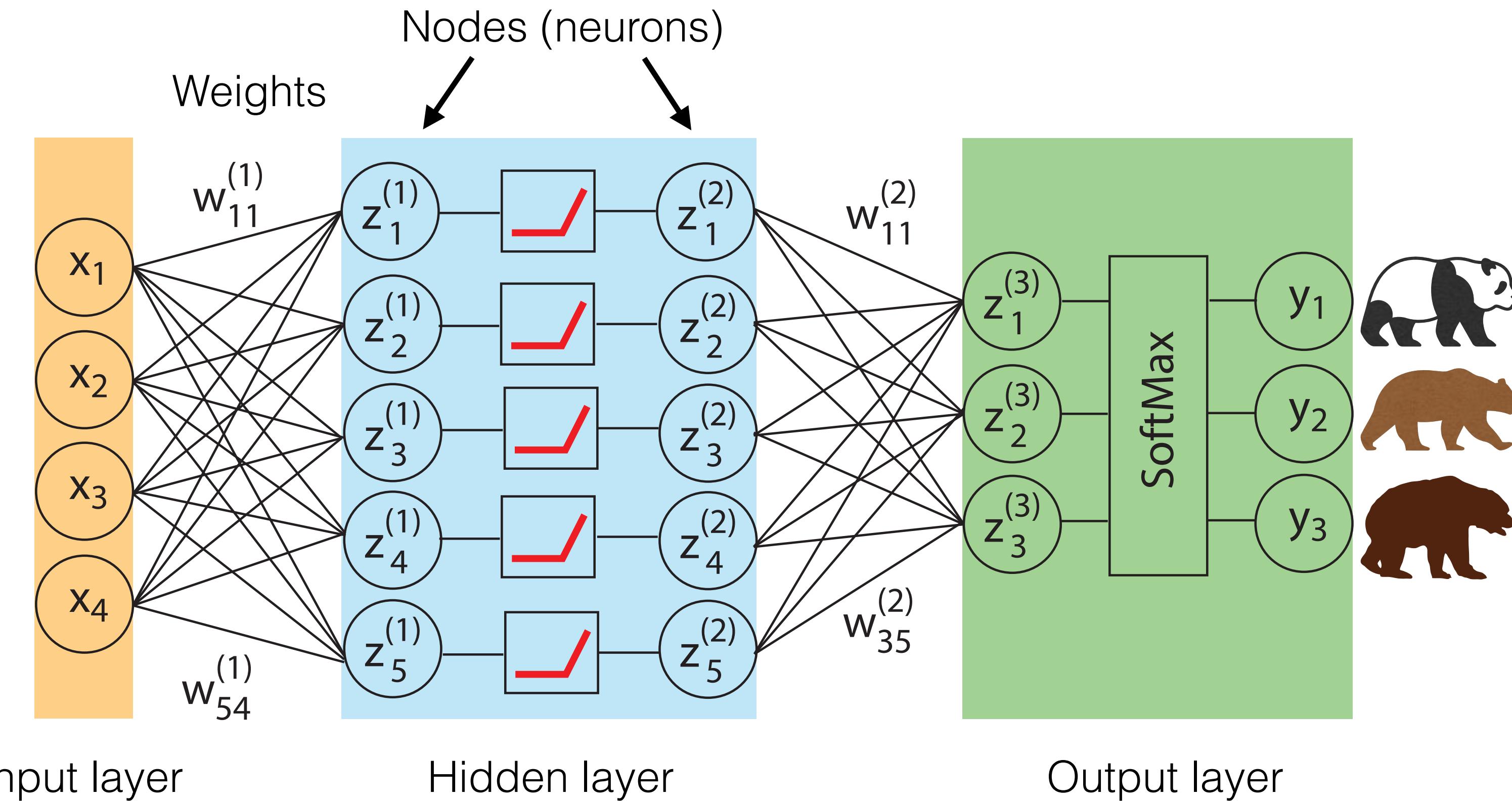
- Intro to neural network model architecture
- An example from biodiversity research
- Neural Network toolbox
- Another applied example and today's tutorial - modeling species richness
- Convolutional neural networks, including applied example

Strengths and weaknesses of neural network models

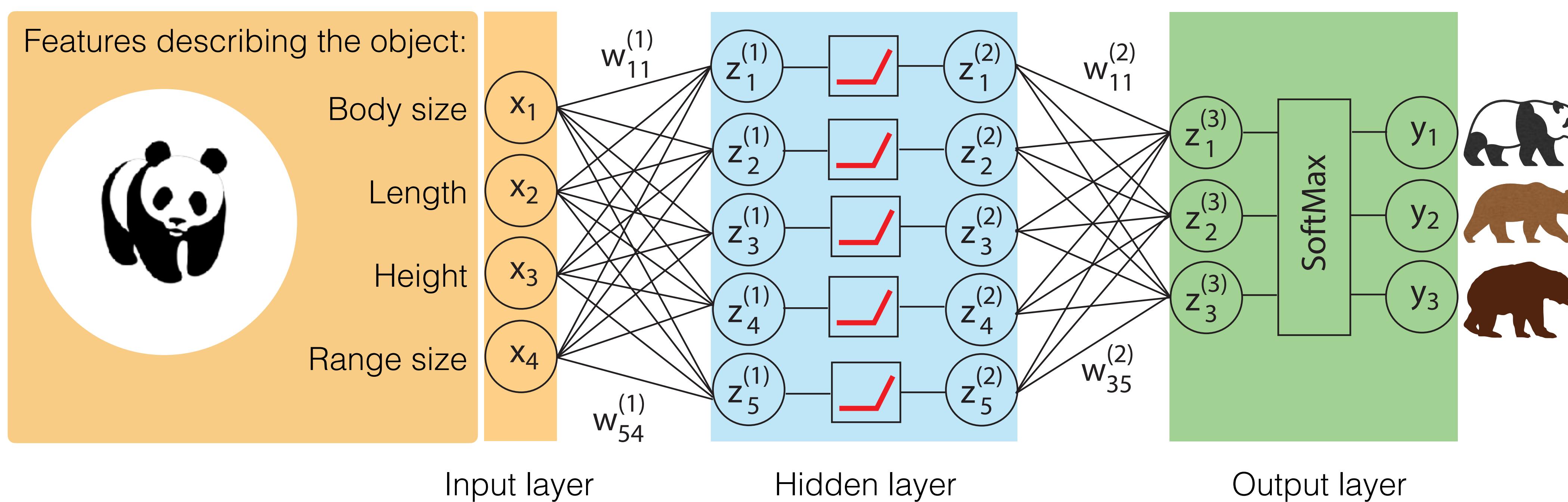


- versatil/flexible
- allow combination of different data types (e.g. images and numerical data)
- model is being learned, does not require defining a specific model (e.g. linear regression) a priori
- can find signal in complex data
- explores intercorrelations and dependencies between different input features
- "black box" model
- not useful if interested in parameter estimates (e.g., correlation coefficient)
- requires substantial training data, not useful for small datasets (>100 instances)
- over-parameterized, simpler models are often better and more interpretable
- care has to be taken so we don't overfit to training data

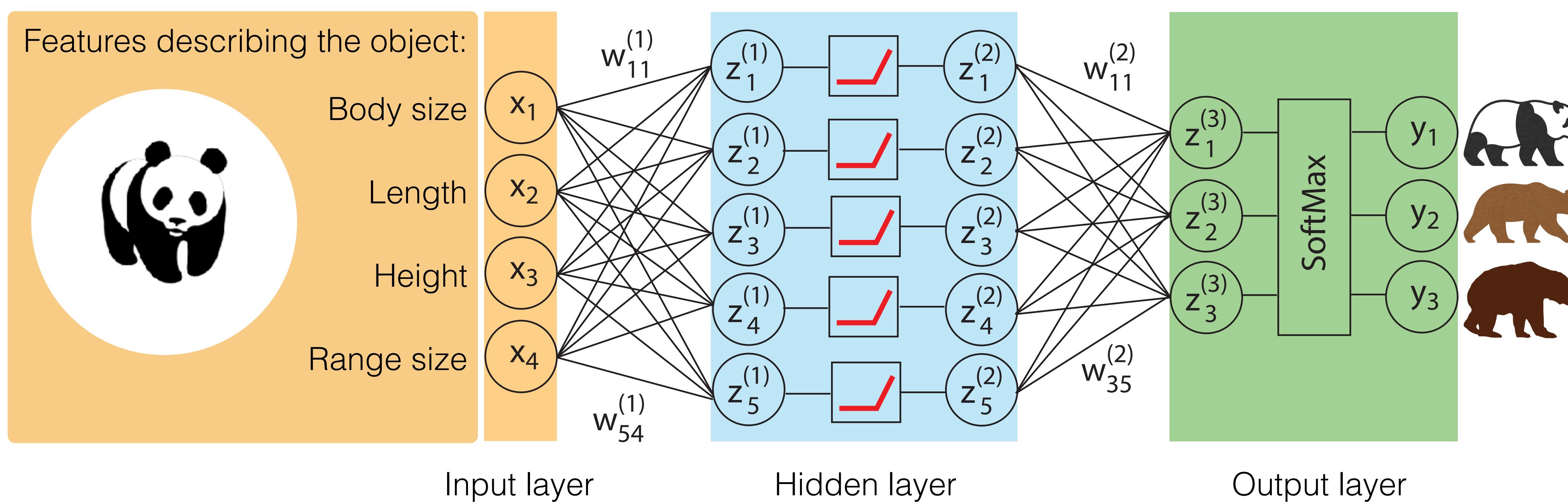
Classification using Neural Networks



Classification using Neural Networks

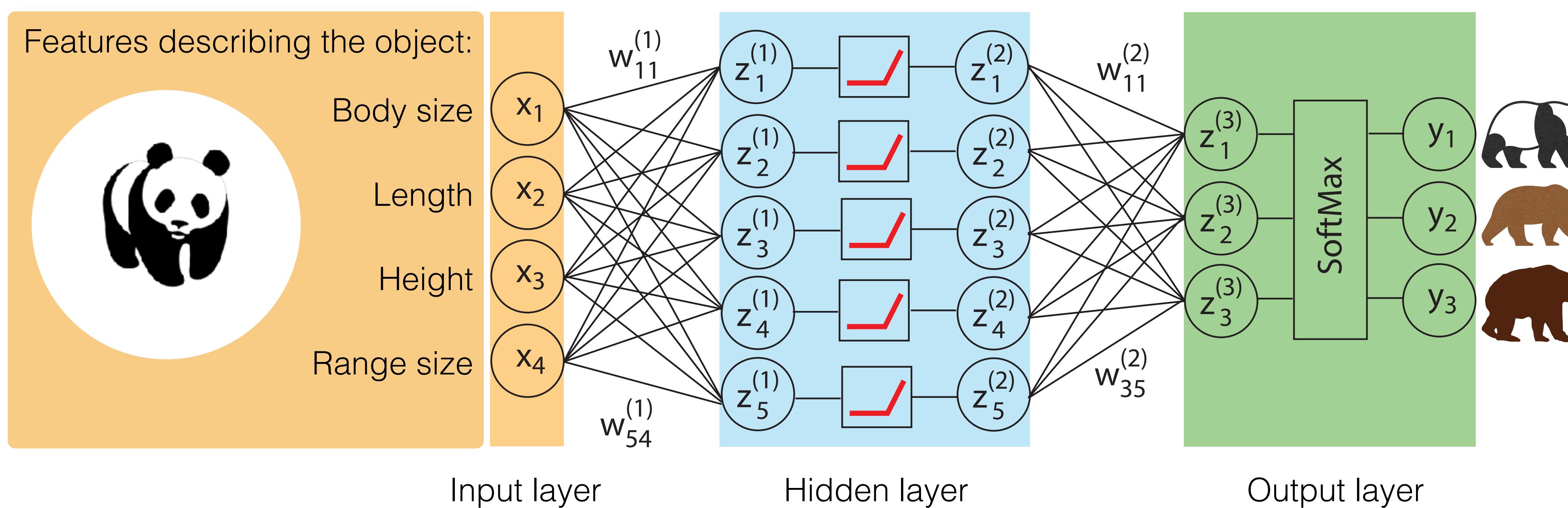


Classification using Neural Networks



Numerical representation of your data in multidimensional space: using weights and activation functions to turn input into output

Classification using Neural Networks



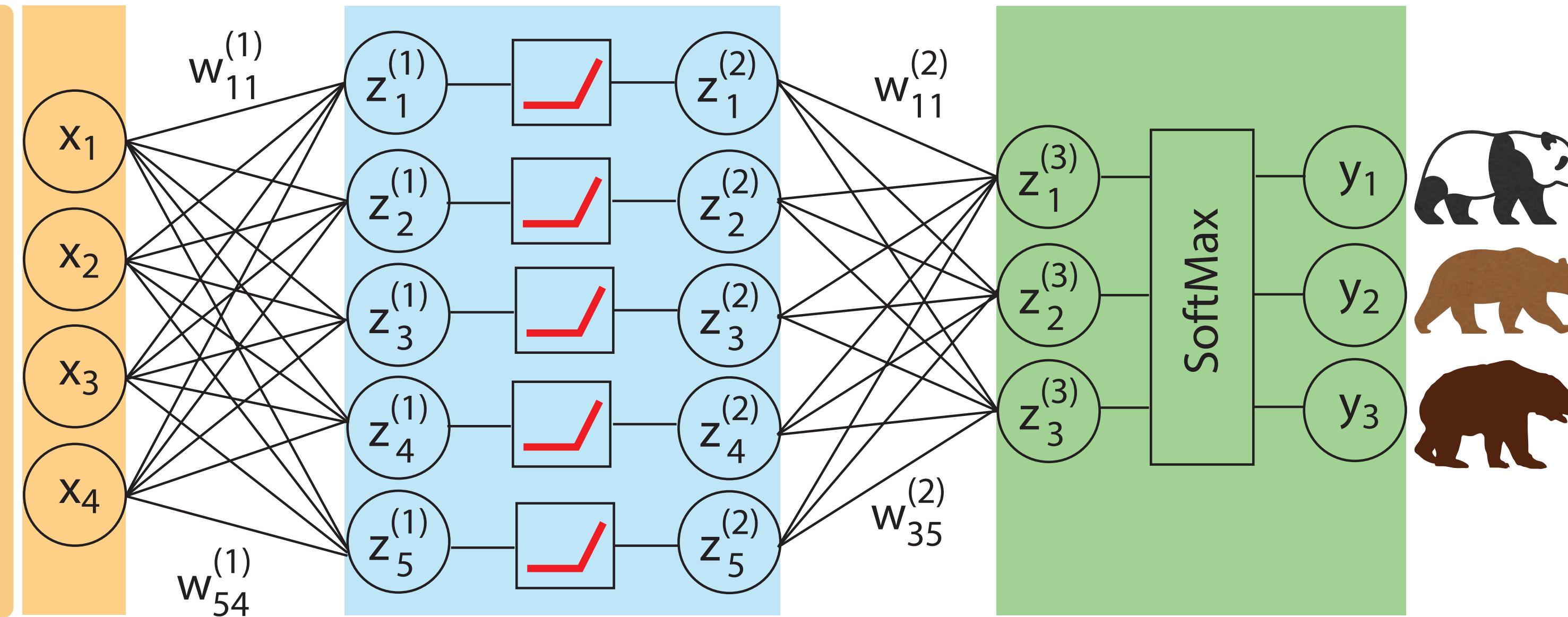
Numerical representation of your data in multidimensional space: using weights and activation functions to turn input into output

Contraction of the hidden layer into a vector (tensor) of the size of the desired output (here 3)

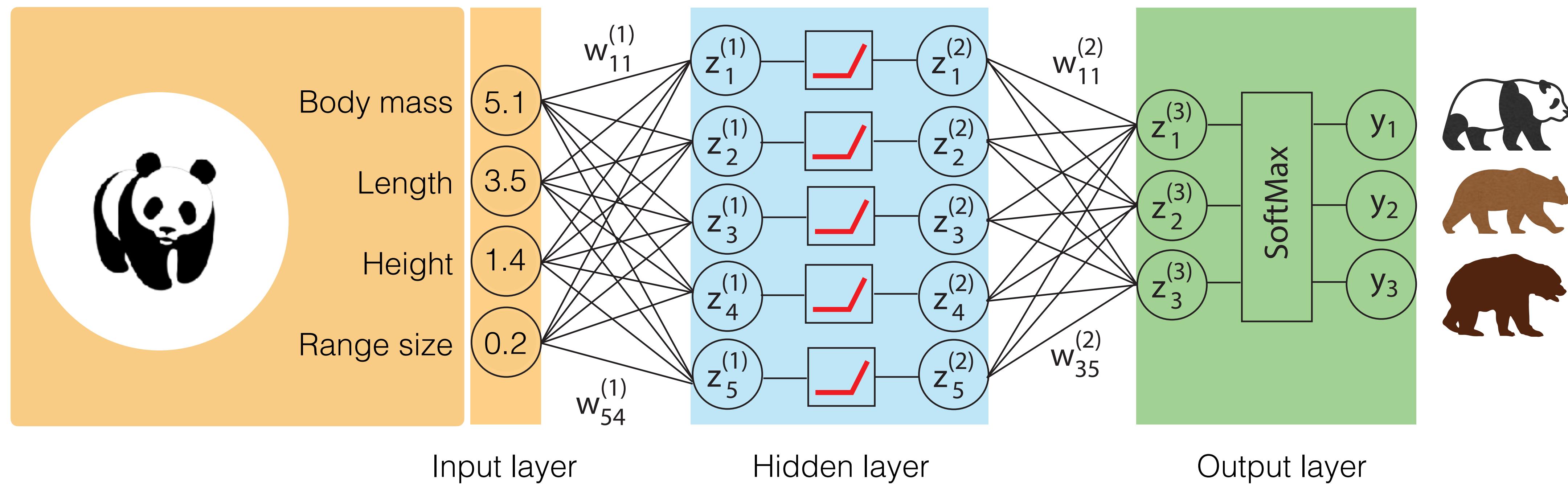
Features describing the object:



Body size
Length
Height
Range size



Forward propagation: from input to output



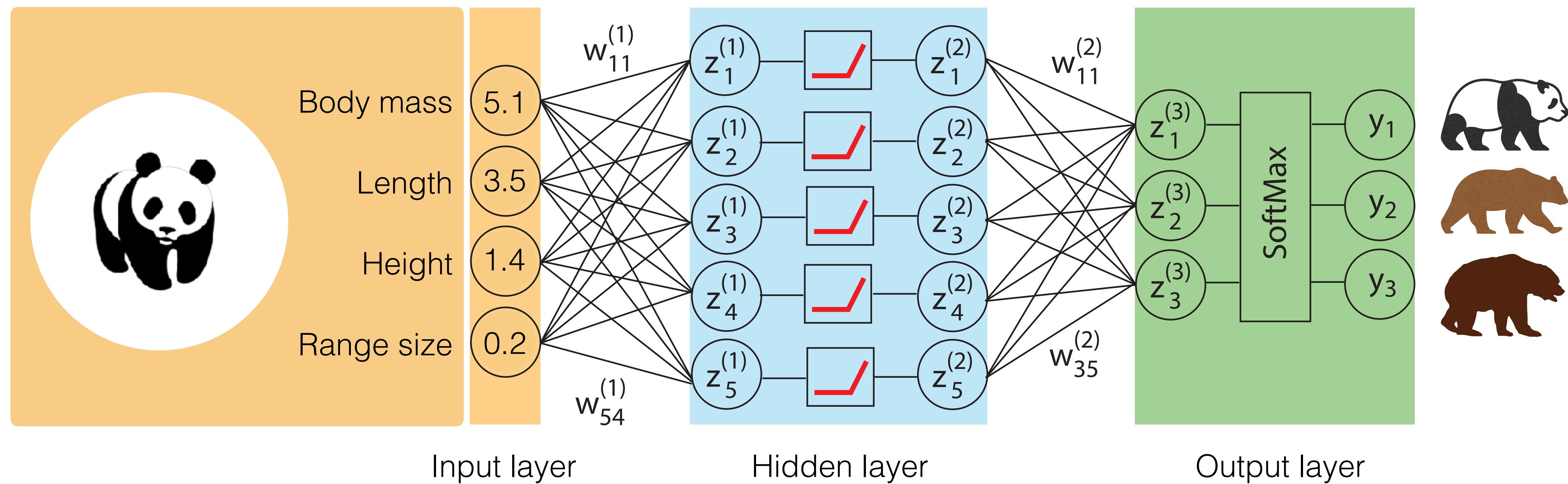
$$\begin{pmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{pmatrix} \begin{pmatrix} 0.5 & 0.1 & -0.2 & -0.4 \\ -0.4 & 1.0 & 0.5 & 1 \\ -0.2 & -0.2 & -0.5 & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix}$$

x

$w^{(1)}$

Matrix multiplication
Multiply each x by w columns
and sum by rows.

Forward propagation: from input to output



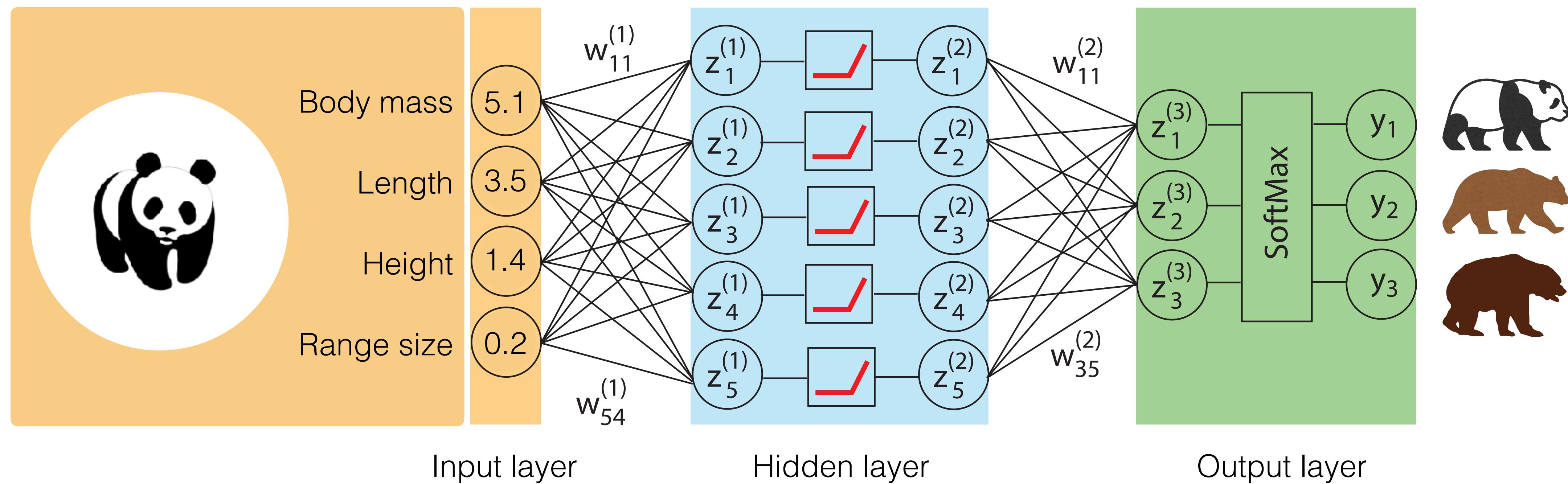
$$\begin{pmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{pmatrix} \begin{pmatrix} 0.5 & -0.2 & -0.4 \\ -0.4 & 1.0 & 0.5 & 1 \\ -0.2 & -0.2 & -0.5 & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix} = \begin{pmatrix} 2.55 & 0.35 & -0.28 & -0.08 \\ -2.04 & 3.5 & 0.7 & 0.2 \\ -1.02 & -0.7 & -0.7 & -0.02 \\ 1.02 & 2.45 & 0.42 & 0.04 \\ 3.06 & 2.1 & 0.14 & -0.08 \end{pmatrix}$$

x

$w^{(1)}$

Matrix multiplication
Multiply each x by w columns
and sum by rows.

Forward propagation: from input to output



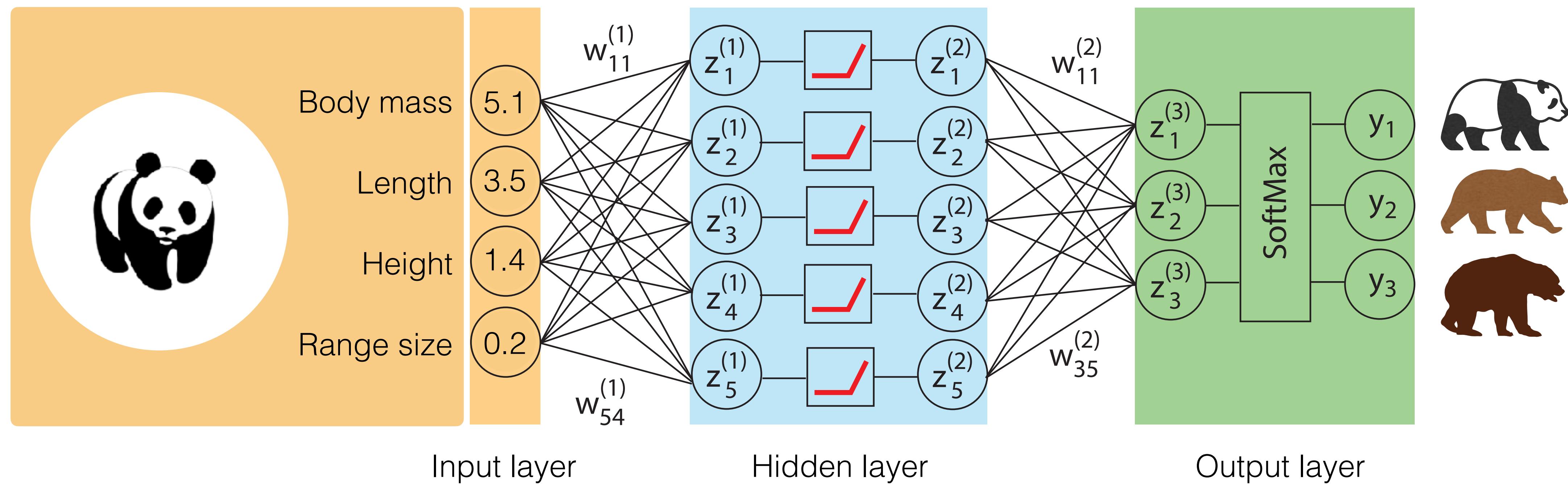
$$\begin{pmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{pmatrix} \begin{pmatrix} 0.5 & 0.1 & -0.2 & -0.4 \\ -0.4 & 1.0 & 0.5 & 1 \\ -0.2 & -0.2 & -0.5 & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix} = \begin{pmatrix} 2.55 & 0.35 & -0.28 & -0.08 \\ -2.04 & 3.5 & 0.7 & 0.2 \\ -1.02 & -0.7 & -0.7 & -0.02 \\ 1.02 & 2.45 & 0.42 & 0.04 \\ 3.06 & 2.1 & 0.14 & -0.08 \end{pmatrix}$$

x

$w^{(1)}$

Matrix multiplication
Multiply each x by w columns
and sum by rows.

Forward propagation: from input to output



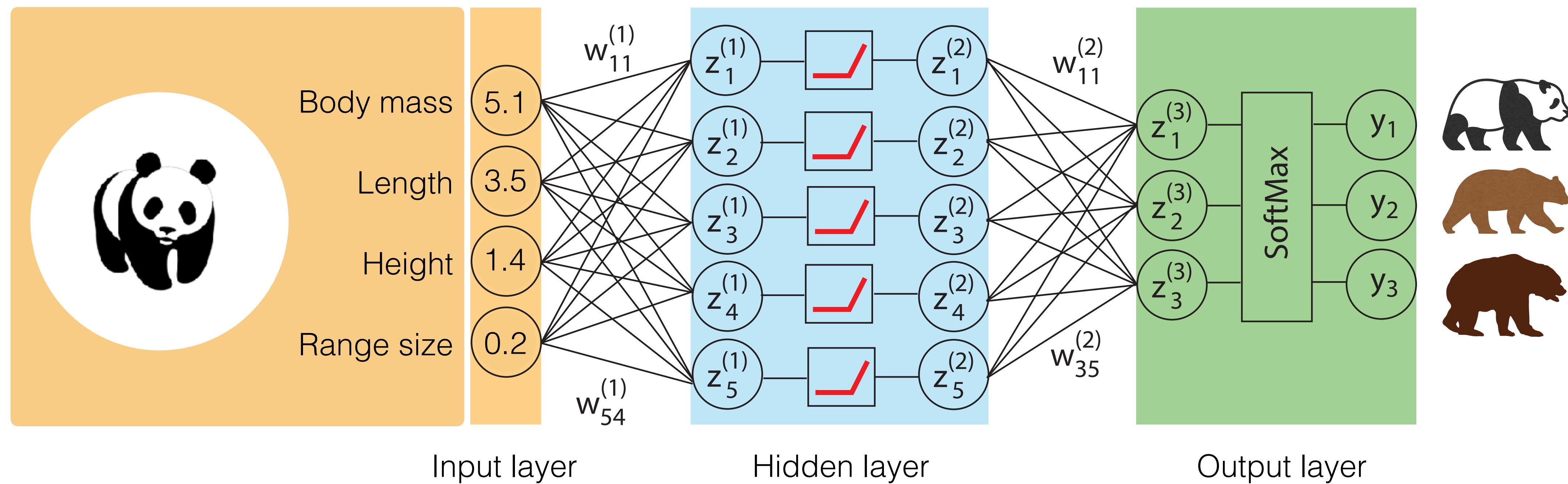
$$\begin{pmatrix} 5.1 \\ 3.5 \\ \boxed{1.4} \\ 0.2 \end{pmatrix} \begin{pmatrix} 0.5 & 0.1 & \boxed{-0.2} & -0.4 \\ -0.4 & 1.0 & 0.5 & 1 \\ -0.2 & -0.2 & \boxed{-0.5} & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix} = \begin{pmatrix} 2.55 & 0.35 & \boxed{-0.28} & -0.08 \\ -2.04 & 3.5 & 0.7 & 0.2 \\ -1.02 & -0.7 & \boxed{-0.7} & -0.02 \\ 1.02 & 2.45 & 0.42 & 0.04 \\ 3.06 & 2.1 & \boxed{0.14} & -0.08 \end{pmatrix}$$

x

$w^{(1)}$

Matrix multiplication
Multiply each x by w columns
and sum by rows.

Forward propagation: from input to output



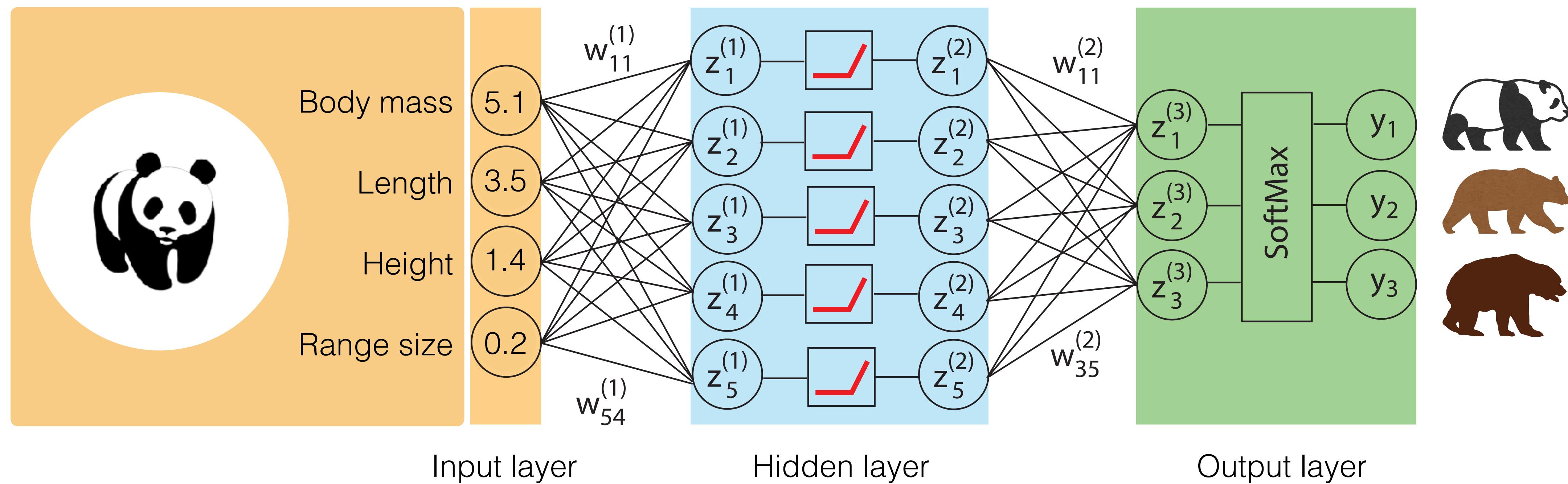
$$\begin{pmatrix} 5.1 \\ 3.5 \\ 1.4 \\ \textcolor{red}{0.2} \end{pmatrix} \begin{pmatrix} 0.5 & 0.1 & -0.2 & \boxed{-0.4} \\ -0.4 & 1.0 & 0.5 & 1 \\ -0.2 & -0.2 & -0.5 & \boxed{-0.1} \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & \boxed{-0.4} \end{pmatrix} = \begin{pmatrix} 2.55 & 0.35 & -0.28 & \boxed{-0.08} \\ -2.04 & 3.5 & 0.7 & 0.2 \\ -1.02 & -0.7 & -0.7 & \boxed{-0.02} \\ 1.02 & 2.45 & 0.42 & 0.04 \\ 3.06 & 2.1 & 0.14 & \boxed{-0.08} \end{pmatrix}$$

x

$w^{(1)}$

Matrix multiplication
Multiply each x by w columns
and sum by rows.

Forward propagation: from input to output



$$\begin{pmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{pmatrix} \begin{pmatrix} 0.5 & 0.1 & -0.2 & -0.4 \\ -0.4 & 1.0 & 0.5 & 1 \\ -0.2 & -0.2 & -0.5 & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix} = \begin{pmatrix} 2.55 & 0.35 & -0.28 & -0.08 \\ -2.04 & 3.5 & 0.7 & 0.2 \\ -1.02 & -0.7 & -0.7 & -0.02 \\ 1.02 & 2.45 & 0.42 & 0.04 \\ 3.06 & 2.1 & 0.14 & -0.08 \end{pmatrix} = \begin{pmatrix} 2.54 \\ 2.36 \\ -2.44 \\ 3.93 \\ 5.22 \end{pmatrix}$$

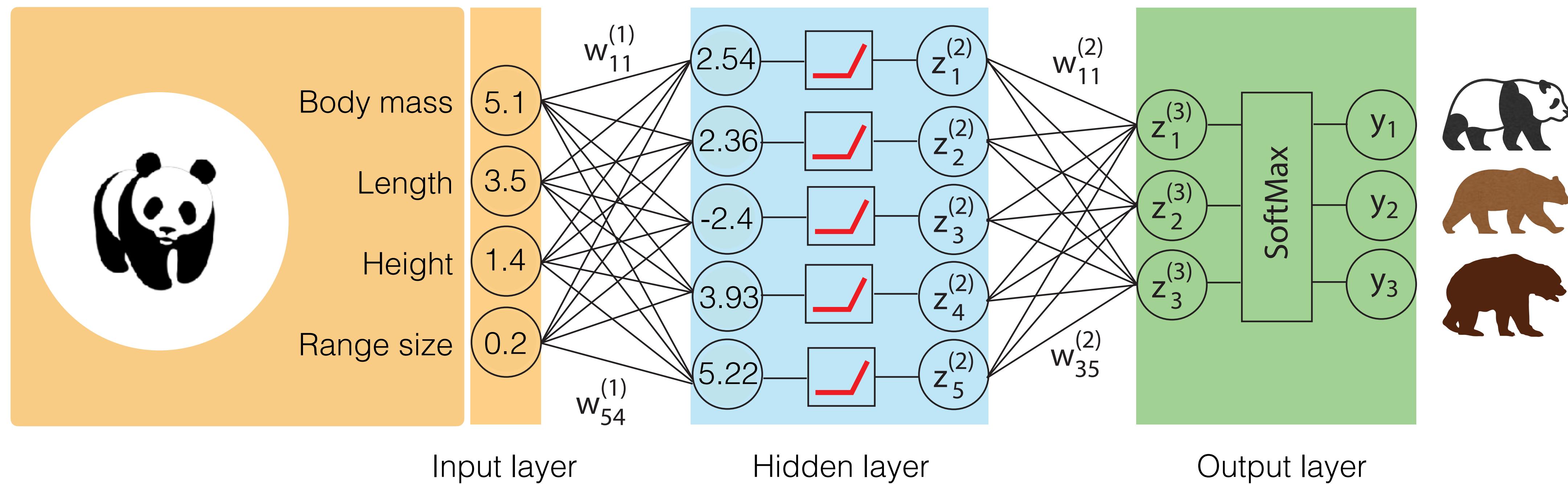
x

$w^{(1)}$

$z^{(1)}$

Matrix multiplication
Multiply each x by w columns
and sum by rows.

Forward propagation: from input to output



$$\begin{pmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{pmatrix} \begin{pmatrix} 0.5 & 0.1 & -0.2 & -0.4 \\ -0.4 & 1.0 & 0.5 & 1 \\ -0.2 & -0.2 & -0.5 & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix} = \begin{pmatrix} 2.55 & 0.35 & -0.28 & -0.08 \\ -2.04 & 3.5 & 0.7 & 0.2 \\ -1.02 & -0.7 & -0.7 & -0.02 \\ 1.02 & 2.45 & 0.42 & 0.04 \\ 3.06 & 2.1 & 0.14 & -0.08 \end{pmatrix} = \begin{pmatrix} 2.54 \\ 2.36 \\ -2.44 \\ 3.93 \\ 5.22 \end{pmatrix}$$

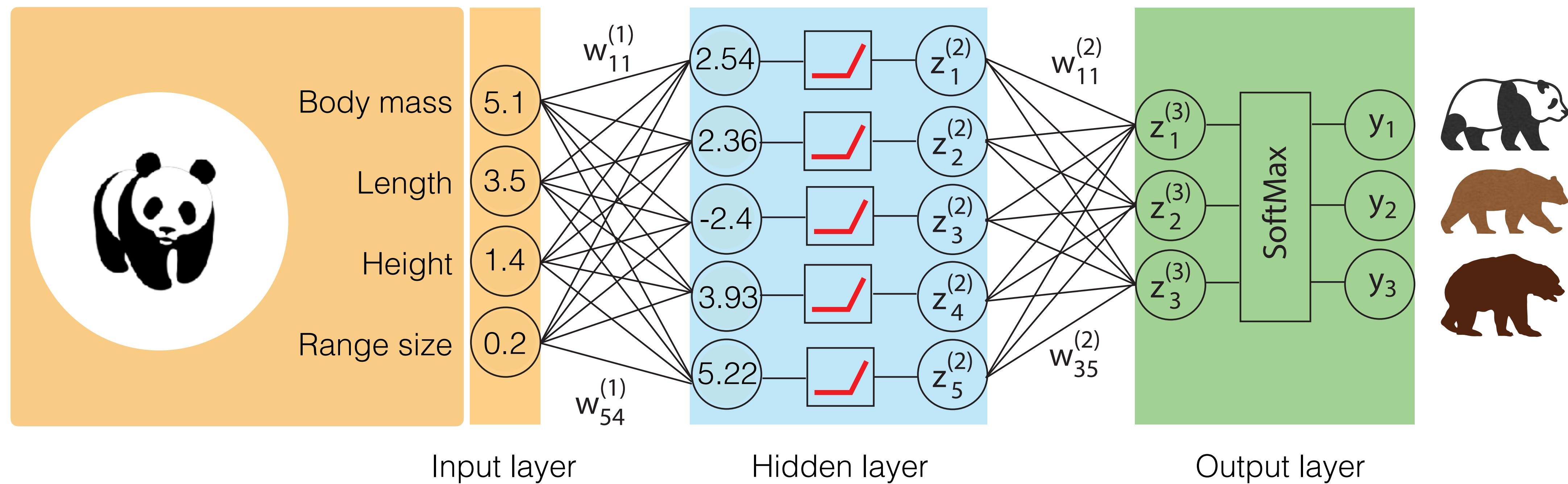
x

$w^{(1)}$

$z^{(1)}$

Matrix multiplication
Multiply each x by w columns
and sum by rows.

Forward propagation: from input to output



$$\begin{pmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{pmatrix} \begin{pmatrix} 0.5 & 0.1 & -0.2 & -0.4 \\ -0.4 & 1.0 & 0.5 & 1 \\ -0.2 & -0.2 & -0.5 & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix} = \begin{pmatrix} 2.55 & 0.35 & -0.28 & -0.08 \\ -2.04 & 3.5 & 0.7 & 0.2 \\ -1.02 & -0.7 & -0.7 & -0.02 \\ 1.02 & 2.45 & 0.42 & 0.04 \\ 3.06 & 2.1 & 0.14 & -0.08 \end{pmatrix} = \begin{pmatrix} 2.54 \\ 2.36 \\ -2.44 \\ 3.93 \\ 5.22 \end{pmatrix}$$

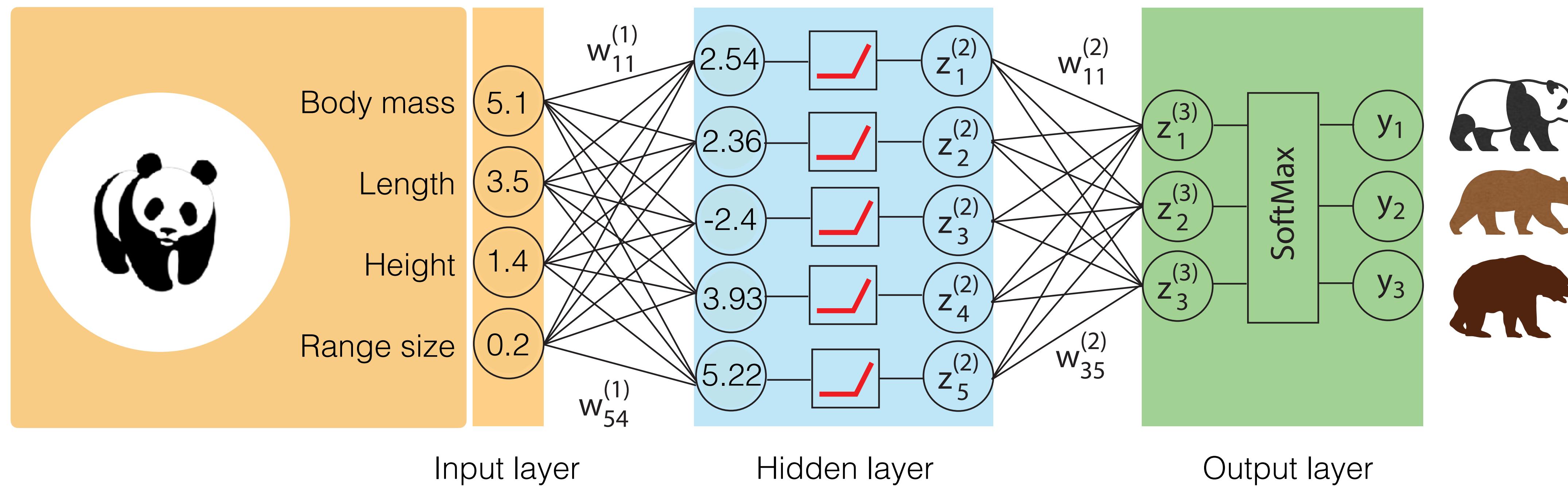
x

$w^{(1)}$

$z^{(1)}$

Matrix multiplication
Multiply each x by w columns
and sum by rows.

Forward propagation: from input to output



$$\begin{pmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{pmatrix} \begin{pmatrix} 0.5 & 0.1 & -0.2 & -0.4 \\ -0.4 & 1.0 & 0.5 & 1 \\ -0.2 & -0.2 & -0.5 & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix} = \begin{pmatrix} 2.54 \\ 2.36 \\ -2.4 \\ 3.93 \\ 5.22 \end{pmatrix}$$

x

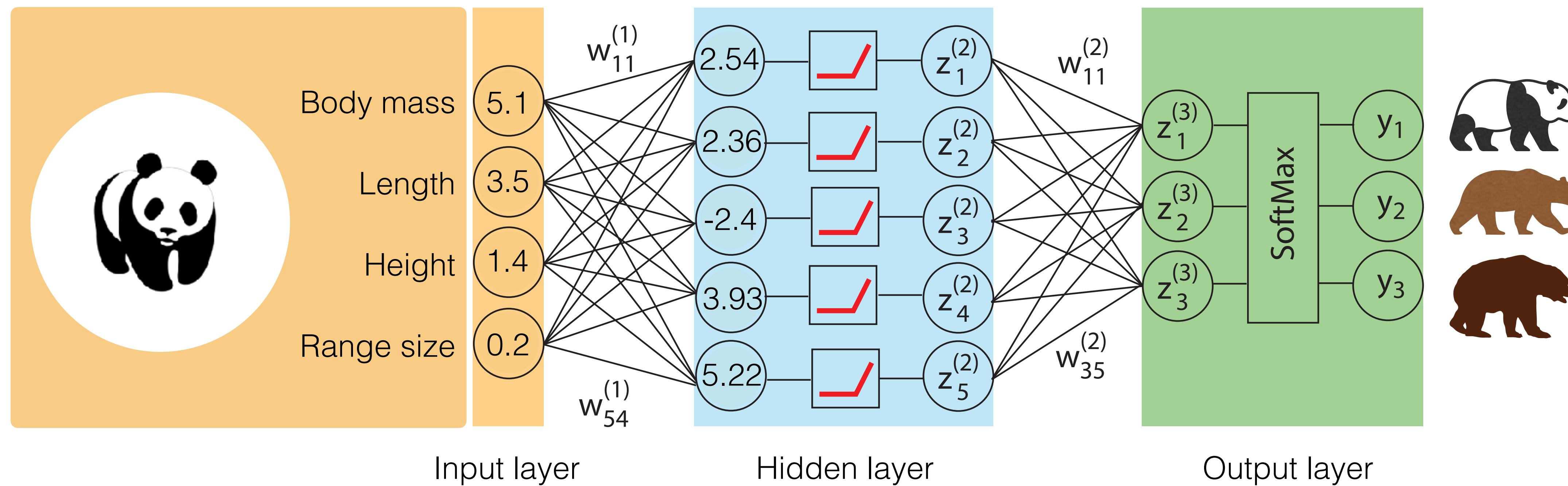
$w^{(1)}$

$z^{(1)}$

Matrix multiplication

Multiply each x by w columns and sum by rows.
This is comfortably computed in 1 line in Python:
 $z1 = np.einsum('j,ij->i', x, w1)$

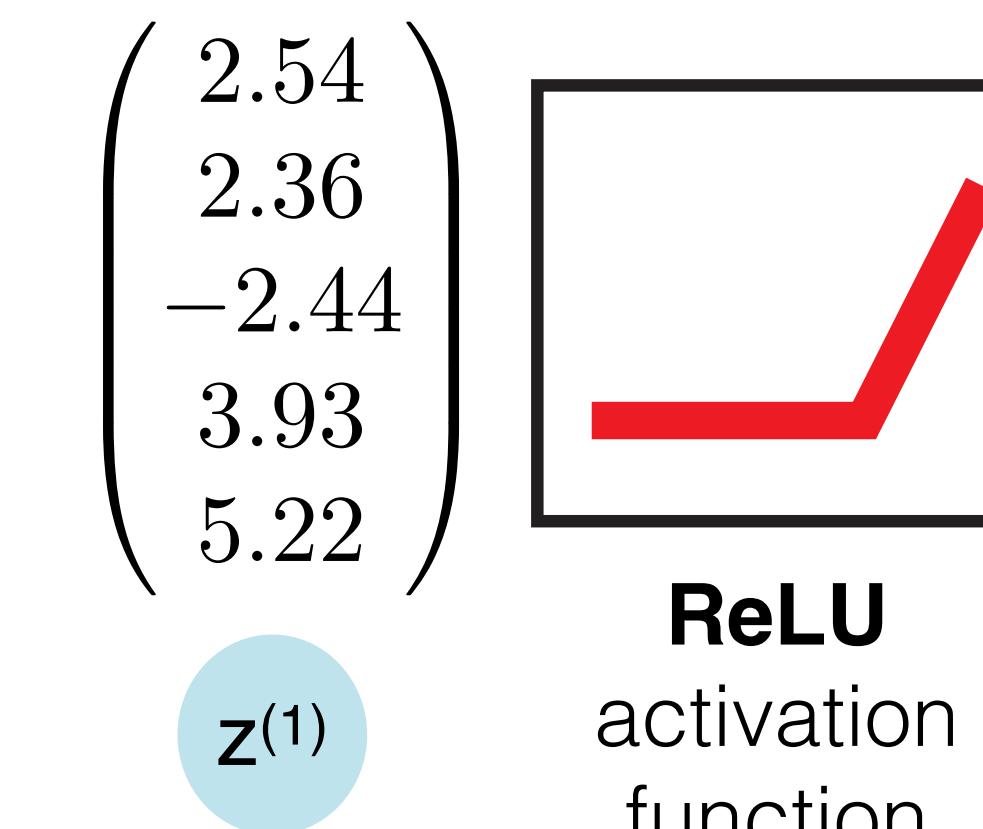
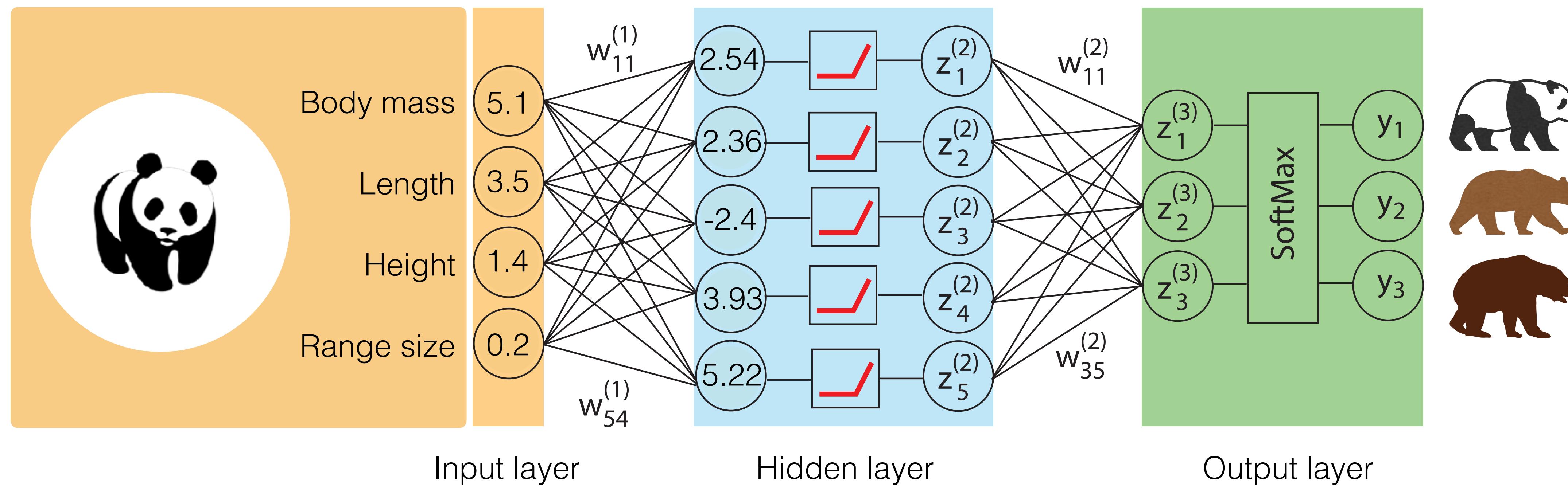
Forward propagation: from input to output



$$\begin{pmatrix} 2.54 \\ 2.36 \\ -2.44 \\ 3.93 \\ 5.22 \end{pmatrix}$$

$z^{(1)}$

Forward propagation: from input to output



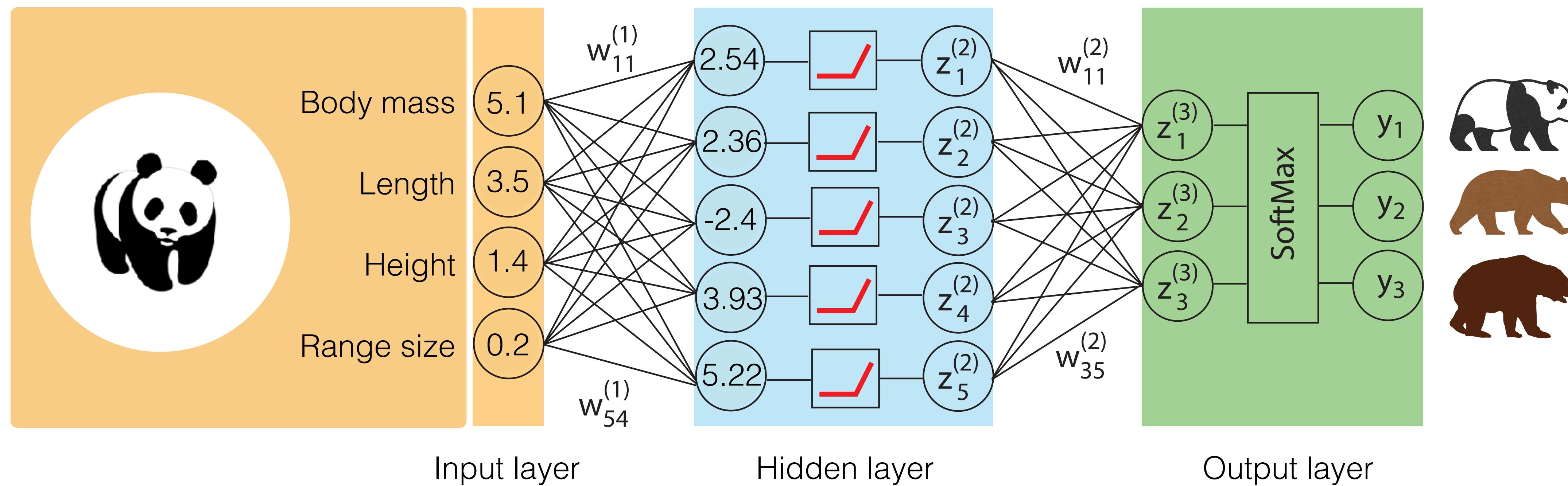
ReLU function

non-linear activation function

negative values are set to 0

$$z_2 = z_1 * (z_1 > 0)$$

Forward propagation: from input to output



$$\begin{pmatrix} 2.54 \\ 2.36 \\ -2.44 \\ 3.93 \\ 5.22 \end{pmatrix} \text{ReLU activation function} = \begin{pmatrix} 2.54 \\ 2.36 \\ 0 \\ 3.93 \\ 5.22 \end{pmatrix}$$

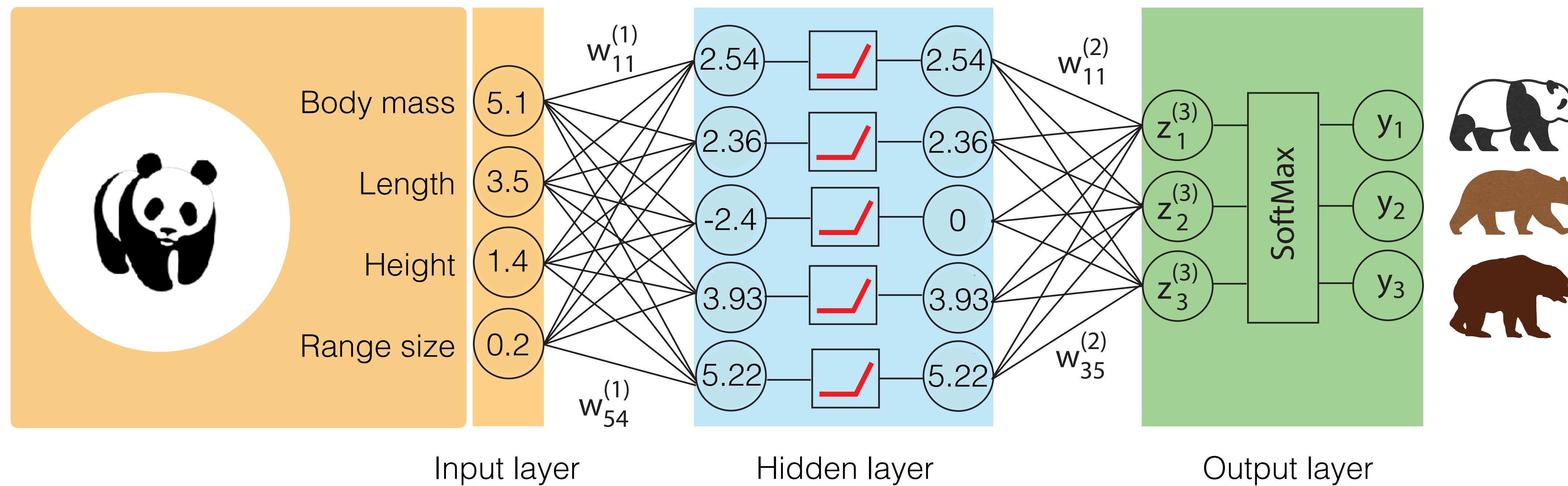
ReLU function

non-linear activation function

negative values are set to 0

$$z_2 = z_1 * (z_1 > 0)$$

Forward propagation: from input to output



$$\begin{pmatrix} 2.54 \\ 2.36 \\ -2.44 \\ 3.93 \\ 5.22 \end{pmatrix} \text{ReLU activation function} = \begin{pmatrix} 2.54 \\ 2.36 \\ 0 \\ 3.93 \\ 5.22 \end{pmatrix}$$

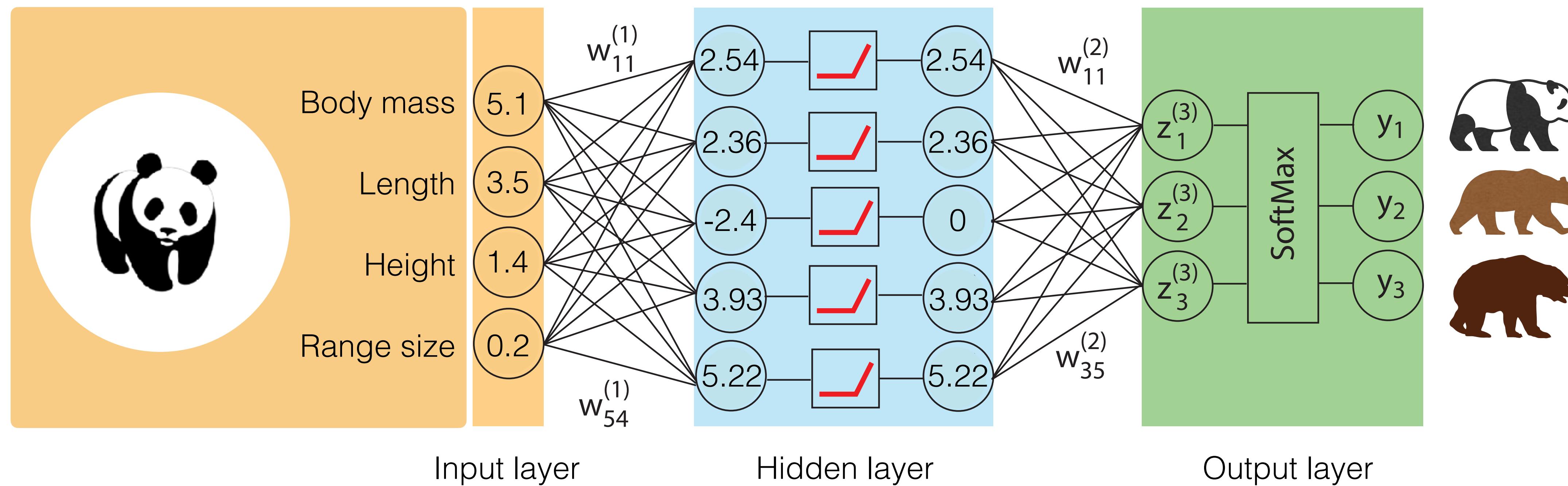
ReLU function

non-linear activation function

negative values are set to 0

$$z_2 = z_1 * (z_1 > 0)$$

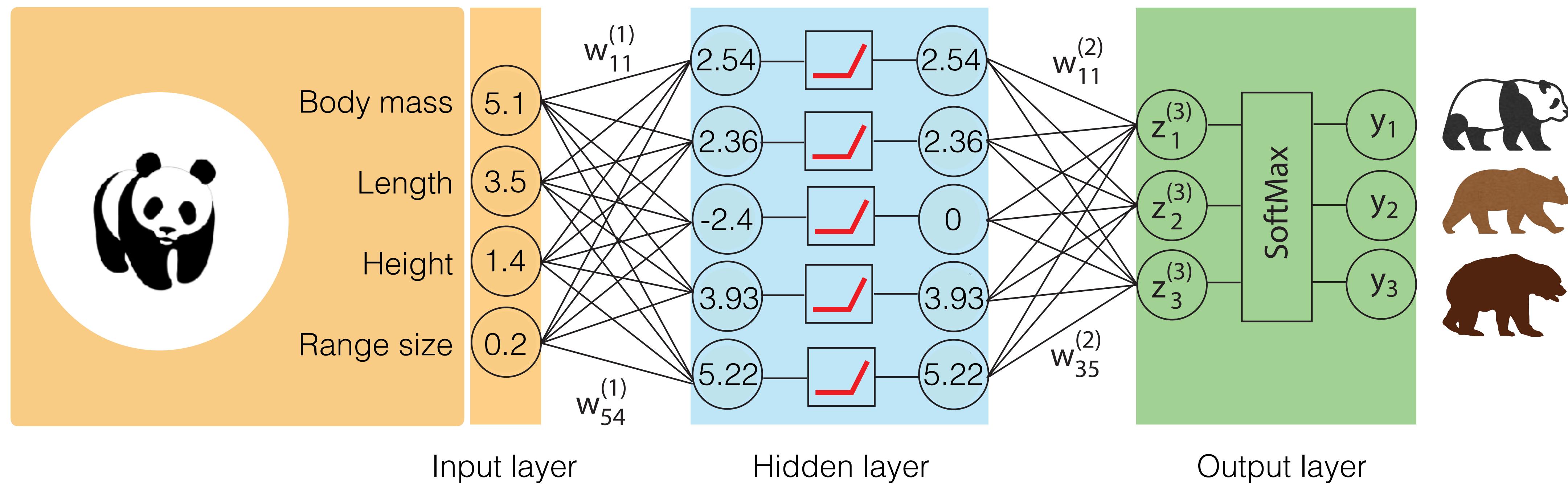
Forward propagation: from input to output



$$\begin{pmatrix} 2.54 \\ 2.36 \\ 0 \\ 3.93 \\ 5.22 \end{pmatrix} \begin{pmatrix} 0.6 & 0.1 & 0.9 & -0.2 & -0.5 \\ 0.3 & -0.3 & 0.3 & -0.9 & -0.9 \\ 0.3 & 0.2 & 0.4 & -1.0 & 0.6 \end{pmatrix}$$

$z^{(2)}$ $w^{(2)}$

Forward propagation: from input to output



Contraction into desired output shape

Matrix multiplication in Python:

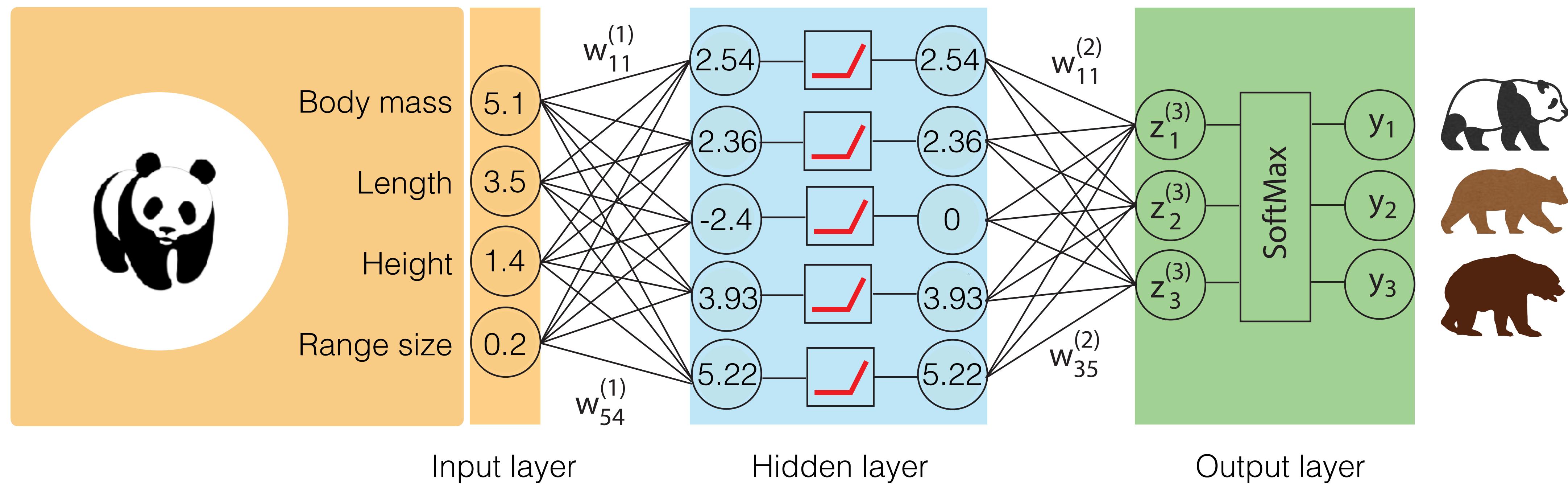
```
z3 = np.einsum('j,ij->i', z2, w2)
```

$$\begin{pmatrix} 2.54 \\ 2.36 \\ 0 \\ 3.93 \\ 5.22 \end{pmatrix} \begin{pmatrix} 0.6 & 0.1 & 0.9 & -0.2 & -0.5 \\ 0.3 & -0.3 & 0.3 & -0.9 & -0.9 \\ 0.3 & 0.2 & 0.4 & -1.0 & 0.6 \end{pmatrix}$$

$z^{(2)}$

$w^{(2)}$

Forward propagation: from input to output



Contraction into desired output shape

Matrix multiplication in Python:

```
z3 = np.einsum('j,ij->i', z2, w2)
```

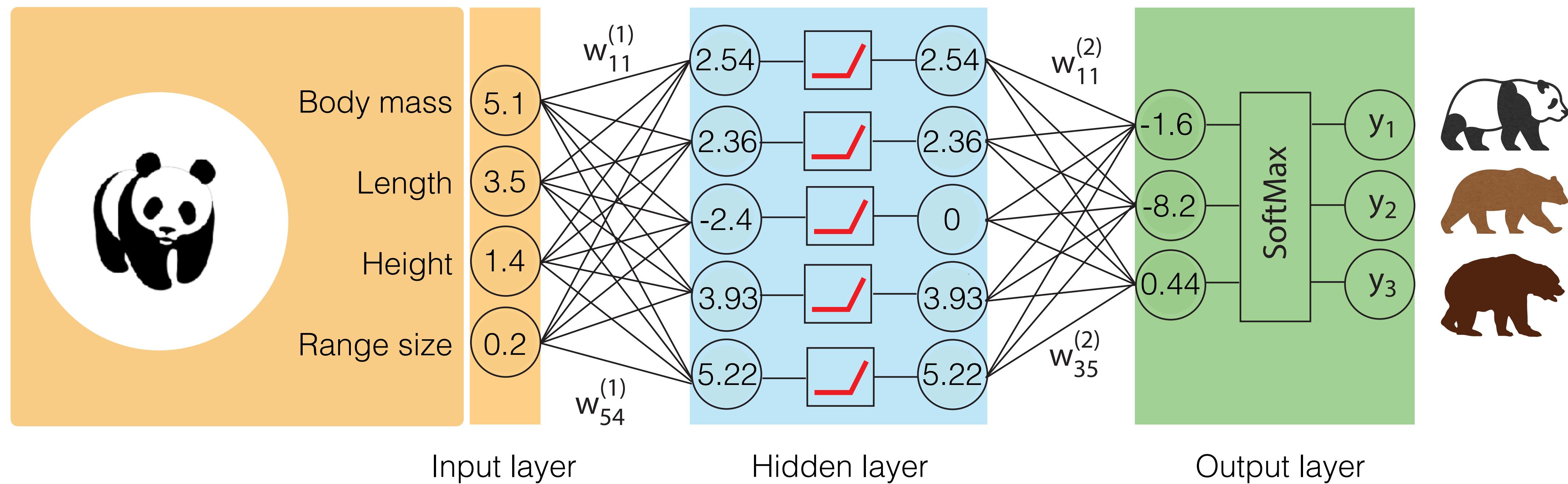
$$\begin{pmatrix} 2.54 \\ 2.36 \\ 0 \\ 3.93 \\ 5.22 \end{pmatrix} \begin{pmatrix} 0.6 & 0.1 & 0.9 & -0.2 & -0.5 \\ 0.3 & -0.3 & 0.3 & -0.9 & -0.9 \\ 0.3 & 0.2 & 0.4 & -1.0 & 0.6 \end{pmatrix} = \begin{pmatrix} -1.64 \\ -8.18 \\ 0.44 \end{pmatrix}$$

$z^{(2)}$

$w^{(2)}$

$z^{(3)}$

Forward propagation: from input to output



Contraction into desired output shape

Matrix multiplication in Python:

```
z3 = np.einsum('j,ij->i', z2, w2)
```

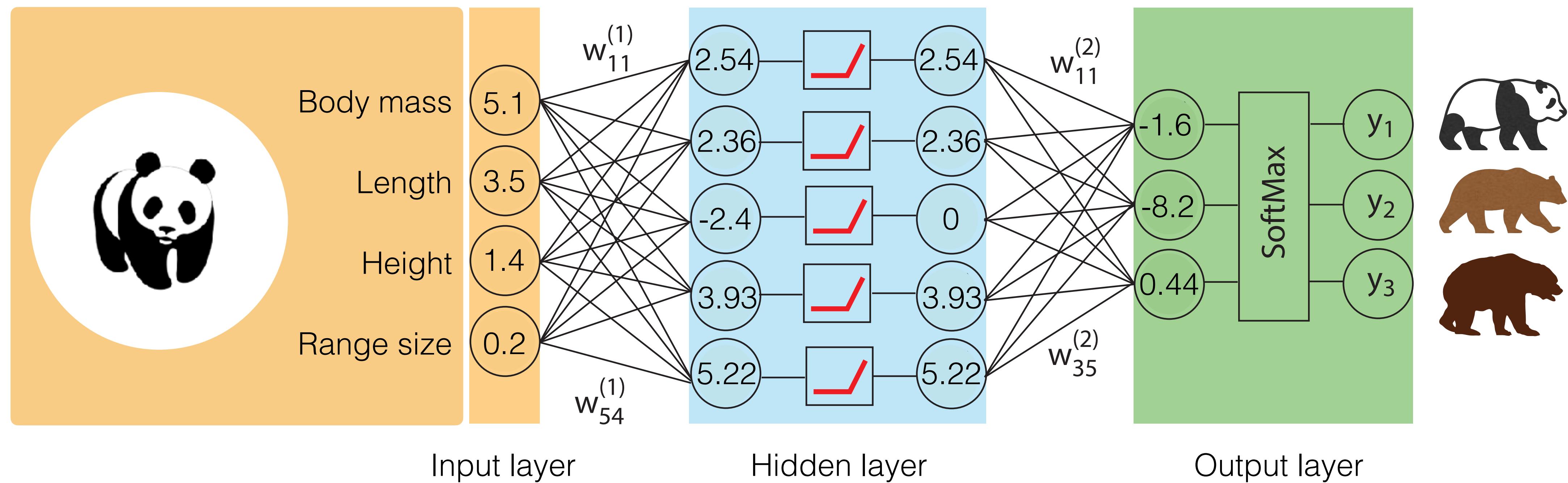
$$\begin{pmatrix} 2.54 \\ 2.36 \\ 0 \\ 3.93 \\ 5.22 \end{pmatrix} \begin{pmatrix} 0.6 & 0.1 & 0.9 & -0.2 & -0.5 \\ 0.3 & -0.3 & 0.3 & -0.9 & -0.9 \\ 0.3 & 0.2 & 0.4 & -1.0 & 0.6 \end{pmatrix} = \begin{pmatrix} -1.64 \\ -8.18 \\ 0.44 \end{pmatrix}$$

$z^{(2)}$

$w^{(2)}$

$z^{(3)}$

Forward propagation: from input to output



SoftMax function

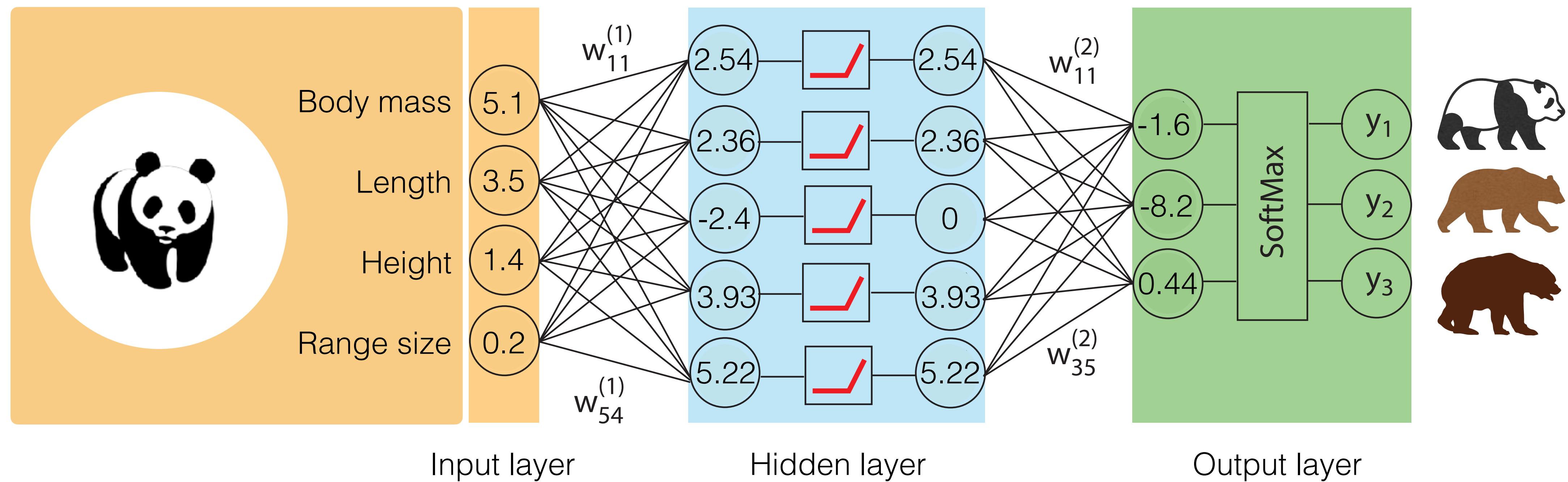
Convert arbitrary real values into probabilities

```
y = np.exp(z3) / np.sum(np.exp(z3))
```

$$\begin{pmatrix} -1.64 \\ -8.18 \\ 0.44 \end{pmatrix}$$

$z^{(3)}$

Forward propagation: from input to output



SoftMax function

Convert arbitrary real values into probabilities

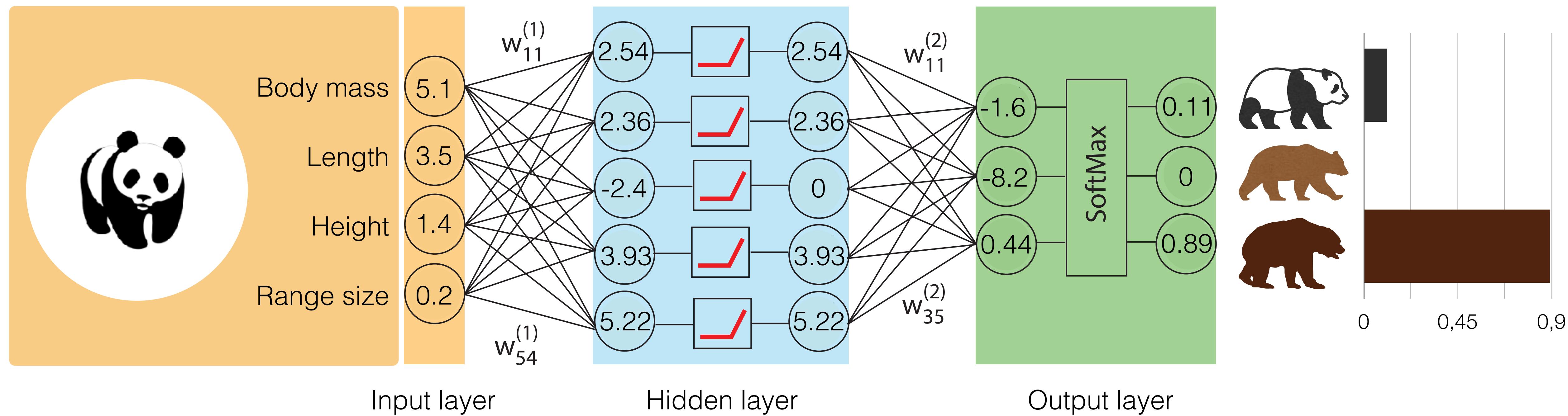
```
y = np.exp(z3) / np.sum(np.exp(z3))
```

$$\begin{aligned}
 \text{SoftMax} &= \frac{e^{-1.64}}{(e^{-1.64} + e^{-8.18} + e^{0.44})} \\
 &\quad \frac{e^{-8.18}}{(e^{-1.64} + e^{-8.18} + e^{0.44})} \\
 &\quad \frac{e^{0.44}}{(e^{-1.64} + e^{-8.18} + e^{0.44})}
 \end{aligned}$$

$z^{(3)}$

$y = \begin{pmatrix} 0.11 \\ 0 \\ 0.89 \end{pmatrix}$

Forward propagation: from input to output



SoftMax function

Convert arbitrary real values into probabilities

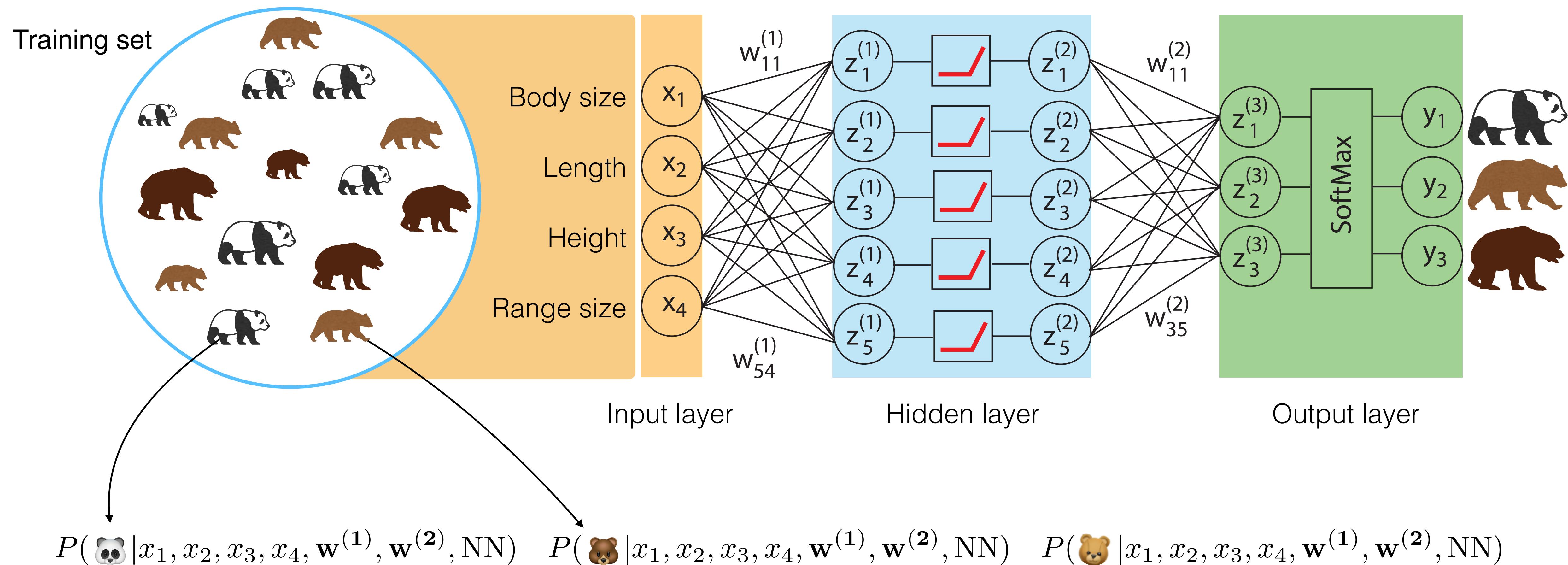
```
y = np.exp(z3) / np.sum(np.exp(z3))
```

$$\begin{aligned}
 \begin{pmatrix} -1.64 \\ -8.18 \\ 0.44 \end{pmatrix} &\rightarrow \frac{e^{-1.64}}{(e^{-1.64} + e^{-8.18} + e^{0.44})} \\
 &\rightarrow \frac{e^{-8.18}}{(e^{-1.64} + e^{-8.18} + e^{0.44})} \\
 &\rightarrow \frac{e^{0.44}}{(e^{-1.64} + e^{-8.18} + e^{0.44})}
 \end{aligned}
 = \begin{pmatrix} 0.11 \\ 0 \\ 0.89 \end{pmatrix}$$

SoftMax

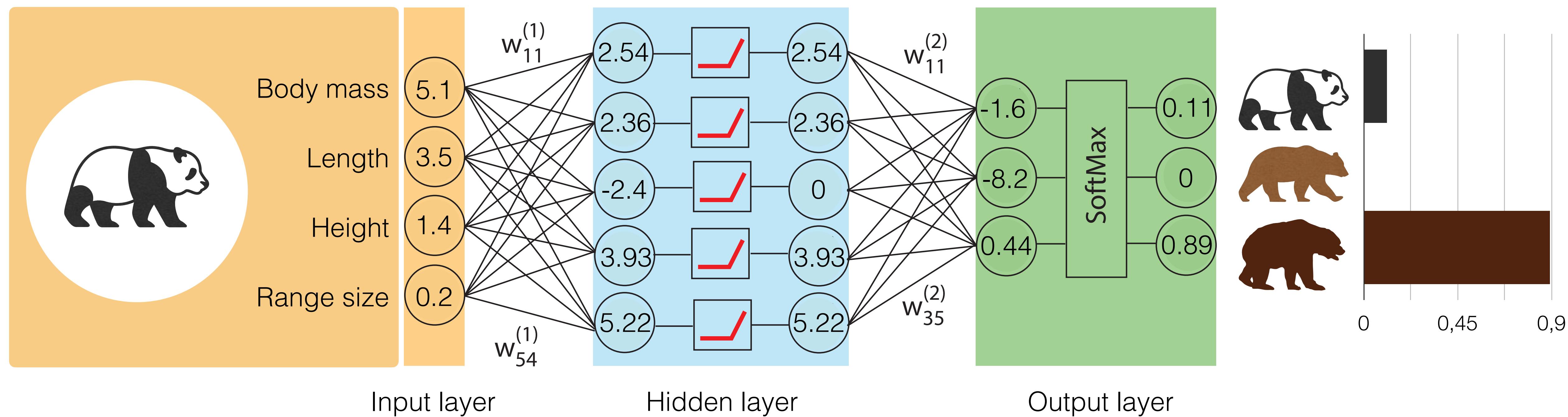
Supervised learning: pre-classified data are used to train the NN model

Training a model means estimating the parameters of the NN model so that the correct output (here the bear species) is predicted based on an input

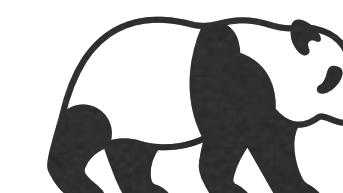


Estimating the parameters (weights) that best map a set of features to the correct class

Supervised learning: minimization of difference between truth and prediction



$$P(\text{🐼} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.11$$



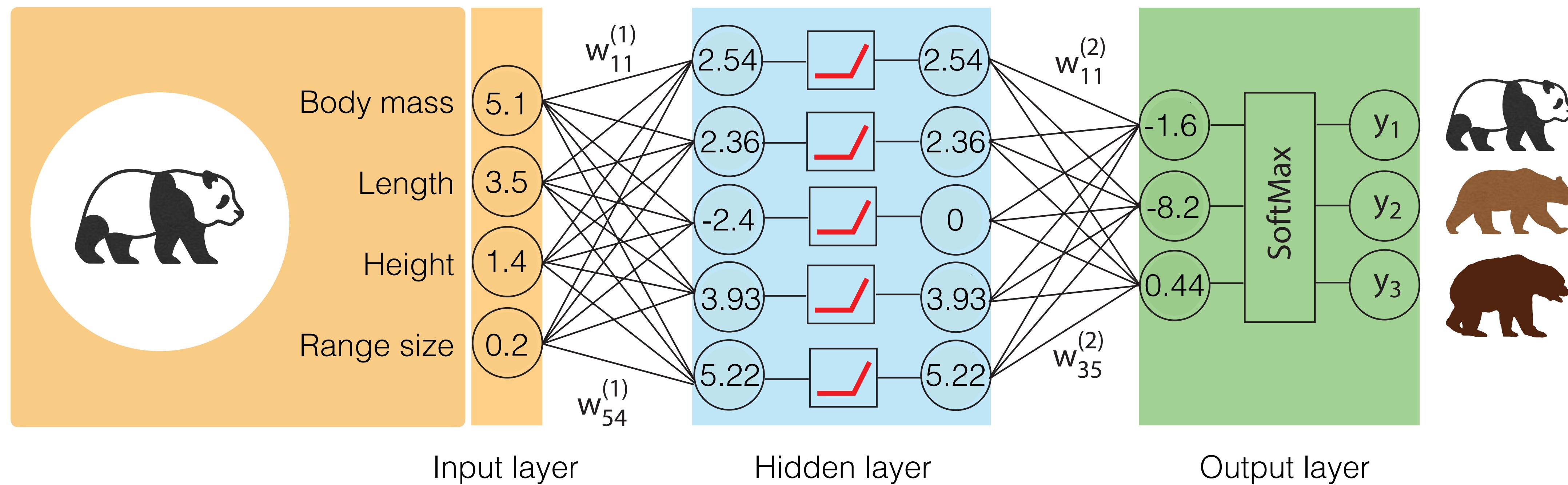
Here we know the features x_1, x_2, x_3, x_4 belong to a panda

$$P(\text{🐻} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0$$

$$P(\text{🐻} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.89 \leftarrow$$

The model with its current parameter values is not optimal because it classified a panda as a brown bear

Training a neural network: minimization of the cross-entropy loss



$$P(\text{🐼} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.11$$

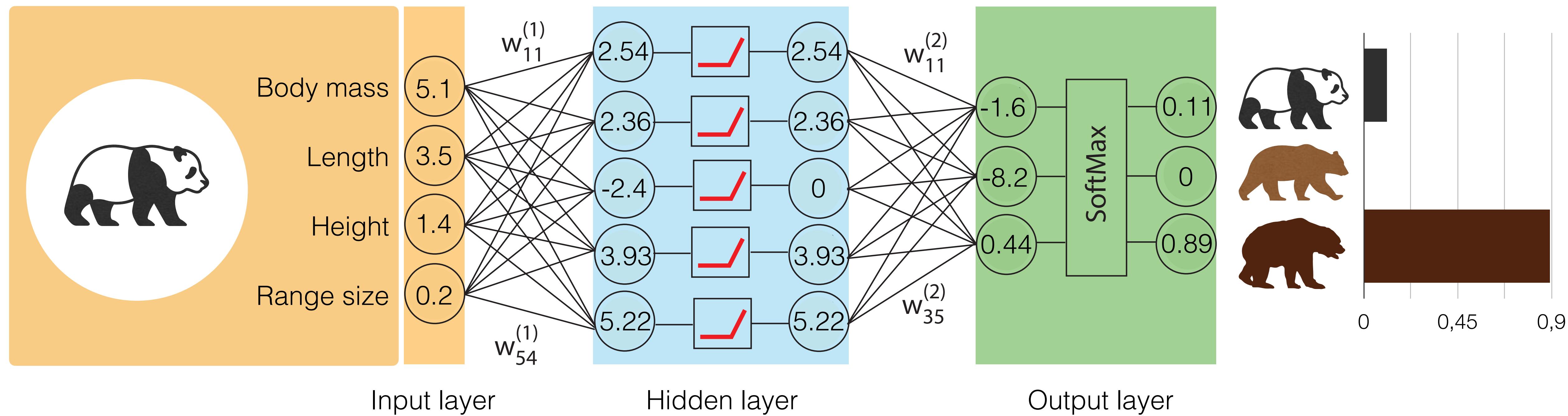
Cross entropy loss: negative log probability of the true class

$$P(\text{🐻} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0$$

$$\text{Loss} = -\log(0.11) = 2.21$$

$$P(\text{🐻} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.89$$

Training a neural network: minimization of the cross-entropy loss



$$P(\text{🐼} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.11$$

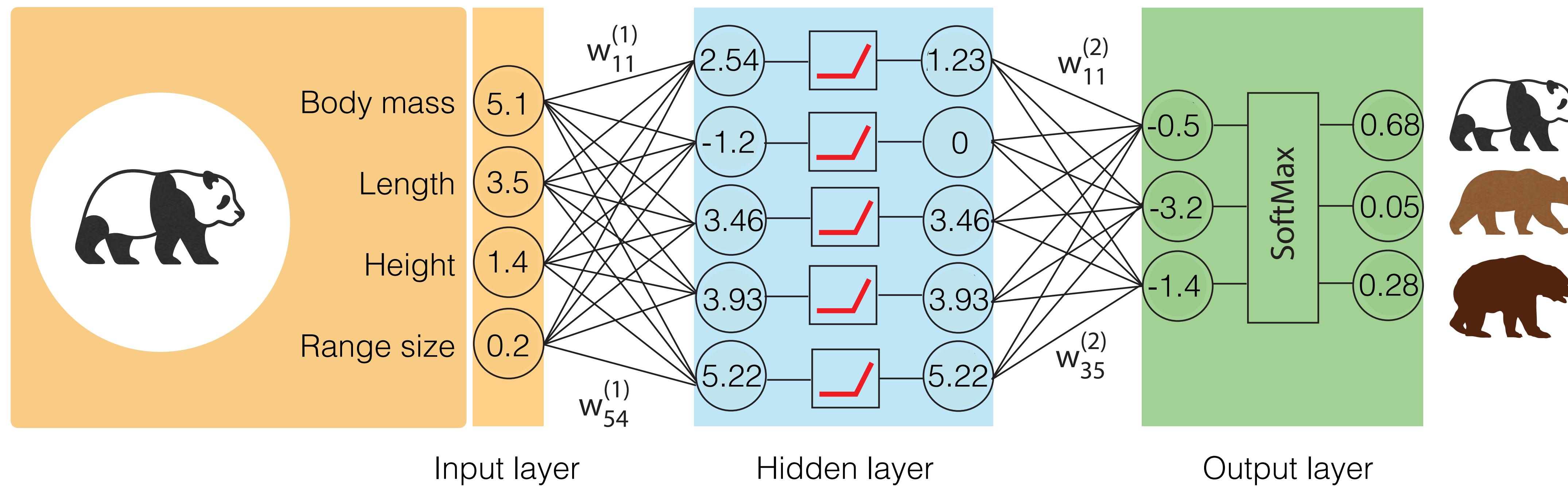
Cross entropy loss: negative log probability of the true class

$$P(\text{🐻} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0$$

$$\text{Loss} = -\log(0.11) = 2.21$$

$$P(\text{🐻} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.89$$

Training a neural network: minimization of the cross-entropy loss



$$P(\text{🐼} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.68$$

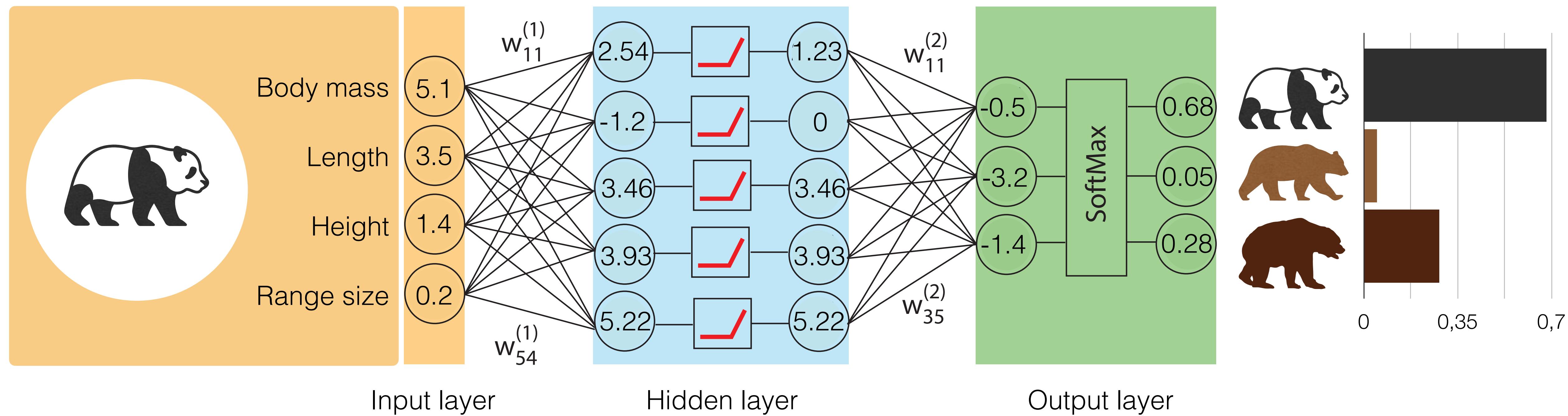
Cross entropy loss: negative log probability of the true class

$$P(\text{🐻} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.05$$

$$\text{Loss} = -\log(0.68) = 0.39$$

$$P(\text{🐻} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.28$$

Training a neural network: minimization of the cross-entropy loss



$$P(\text{🐼} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.68$$

Cross entropy loss: negative log probability of the true class

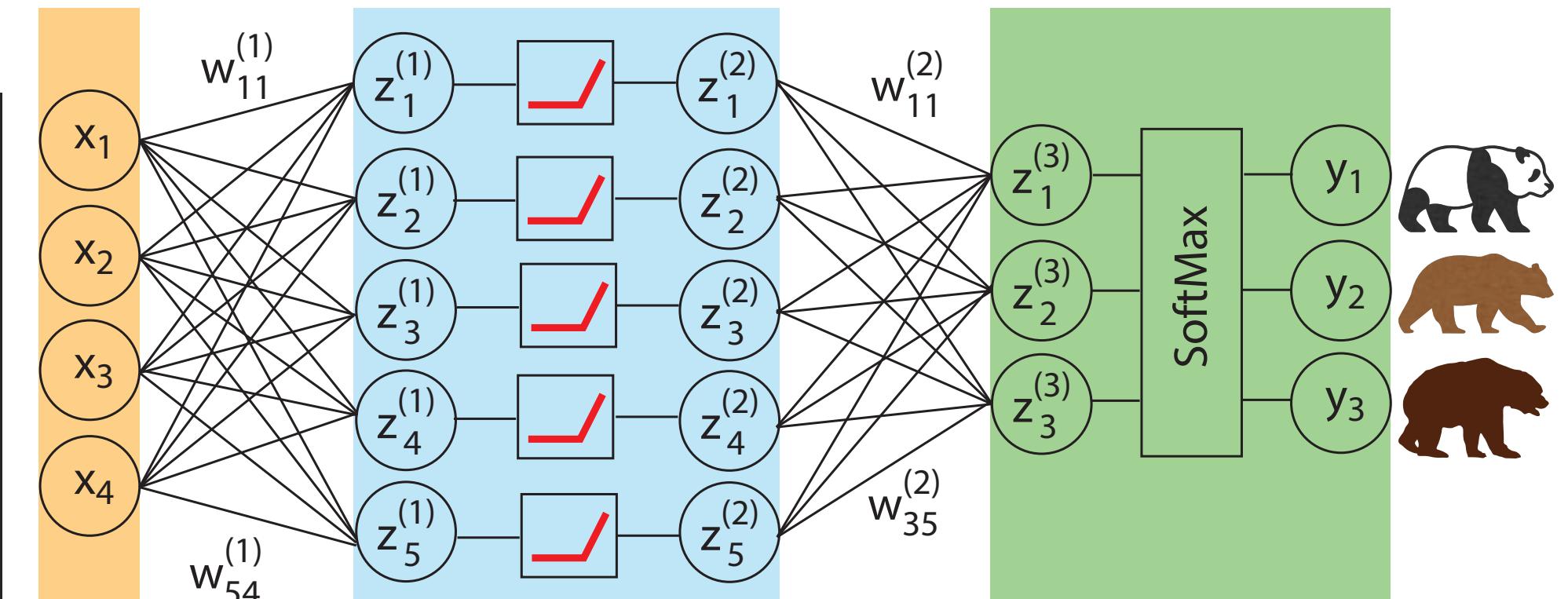
$$P(\text{🐻} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.05$$

$$\text{Loss} = -\log(0.68) = 0.39$$

$$P(\text{🐻} | x_1, x_2, x_3, x_4, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = 0.28$$

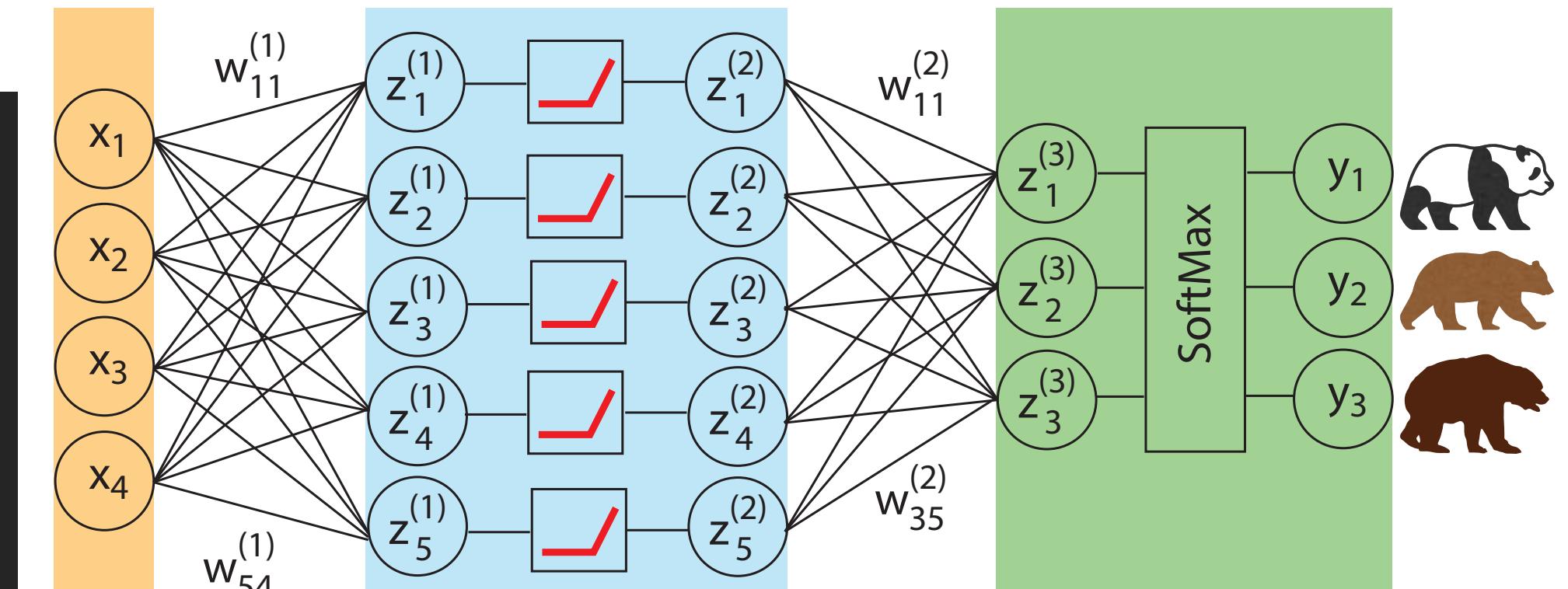
Building neural network model in tensorflow

```
# Example for classification model  
  
architecture = []  
  
# Input layer  
architecture.append(tf.keras.layers.Flatten(input_shape=[4]))  
# 1st hidden layer  
architecture.append(tf.keras.layers.Dense(5, activation='relu'))  
# Output layer  
architecture.append(tf.keras.layers.Dense(3, activation='softmax'))  
  
# Compile the model  
model = tf.keras.Sequential(architecture)  
model.compile(loss='categorical_crossentropy',  
              optimizer='adam', metrics=['accuracy'])  
  
# Get overview of model architecture  
model.summary()
```



Building neural network model in tensorflow

```
# Example for classification model  
  
architecture = []  
  
# Input layer  
architecture.append(tf.keras.layers.Flatten(input_shape=[4]))  
# 1st hidden layer  
architecture.append(tf.keras.layers.Dense(5, activation='relu'))  
# Output layer  
architecture.append(tf.keras.layers.Dense(3, activation='softmax'))  
  
# Compile the model  
model = tf.keras.Sequential(architecture)  
model.compile(loss='categorical_crossentropy',  
              optimizer='adam', metrics=['accuracy'])  
  
# Get overview of model architecture  
model.summary()
```



```
>>> model.summary()  
Model: "sequential_9"  
-----  
Layer (type)          Output Shape         Param #  
-----  
flatten_9 (Flatten)    (None, 4)           0  
  
dense_21 (Dense)      (None, 5)           20  
-----  
dense_22 (Dense)      (None, 5)           25  
-----  
Total params: 45  
Trainable params: 45  
Non-trainable params: 0  
-----
```

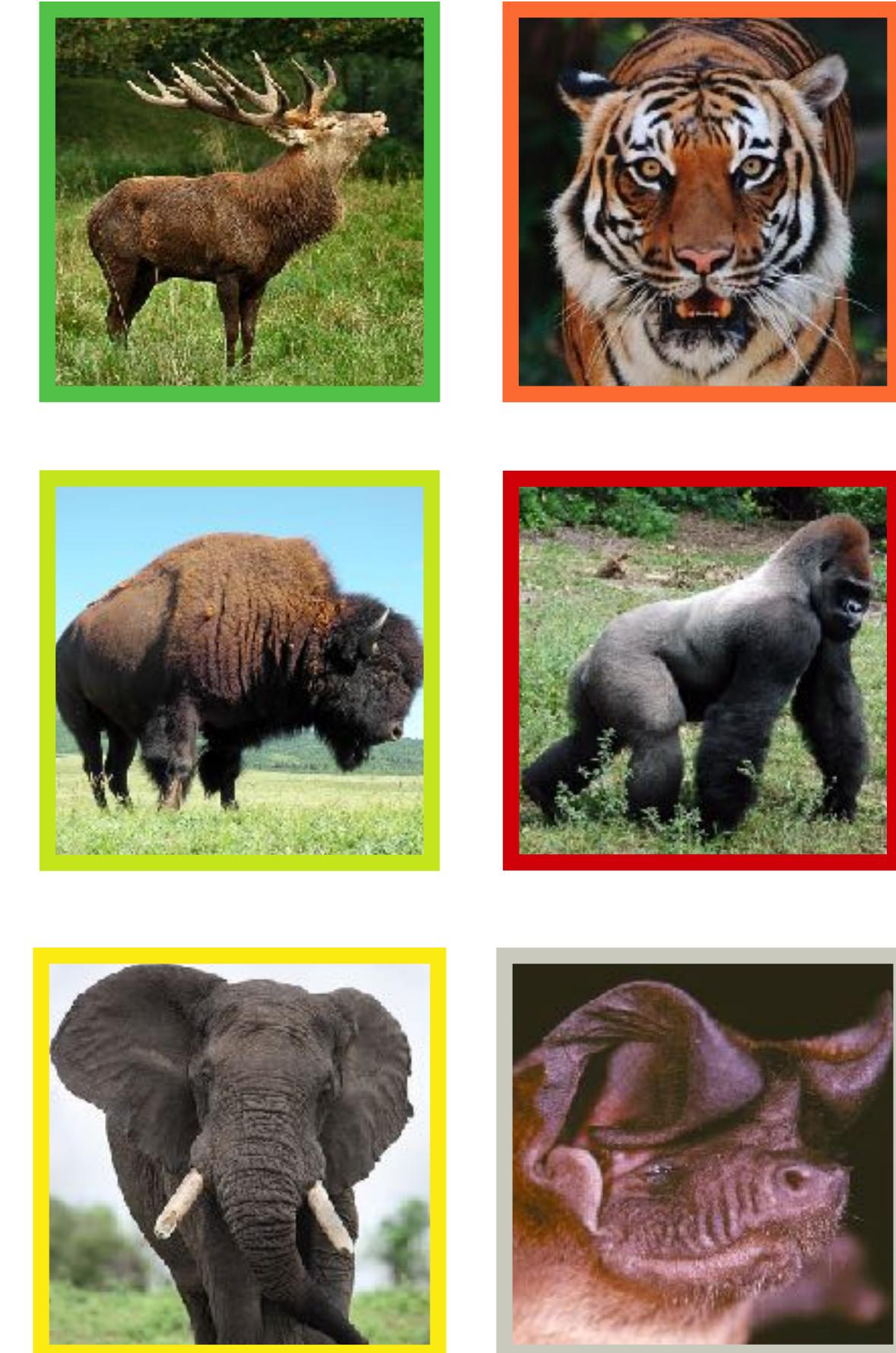
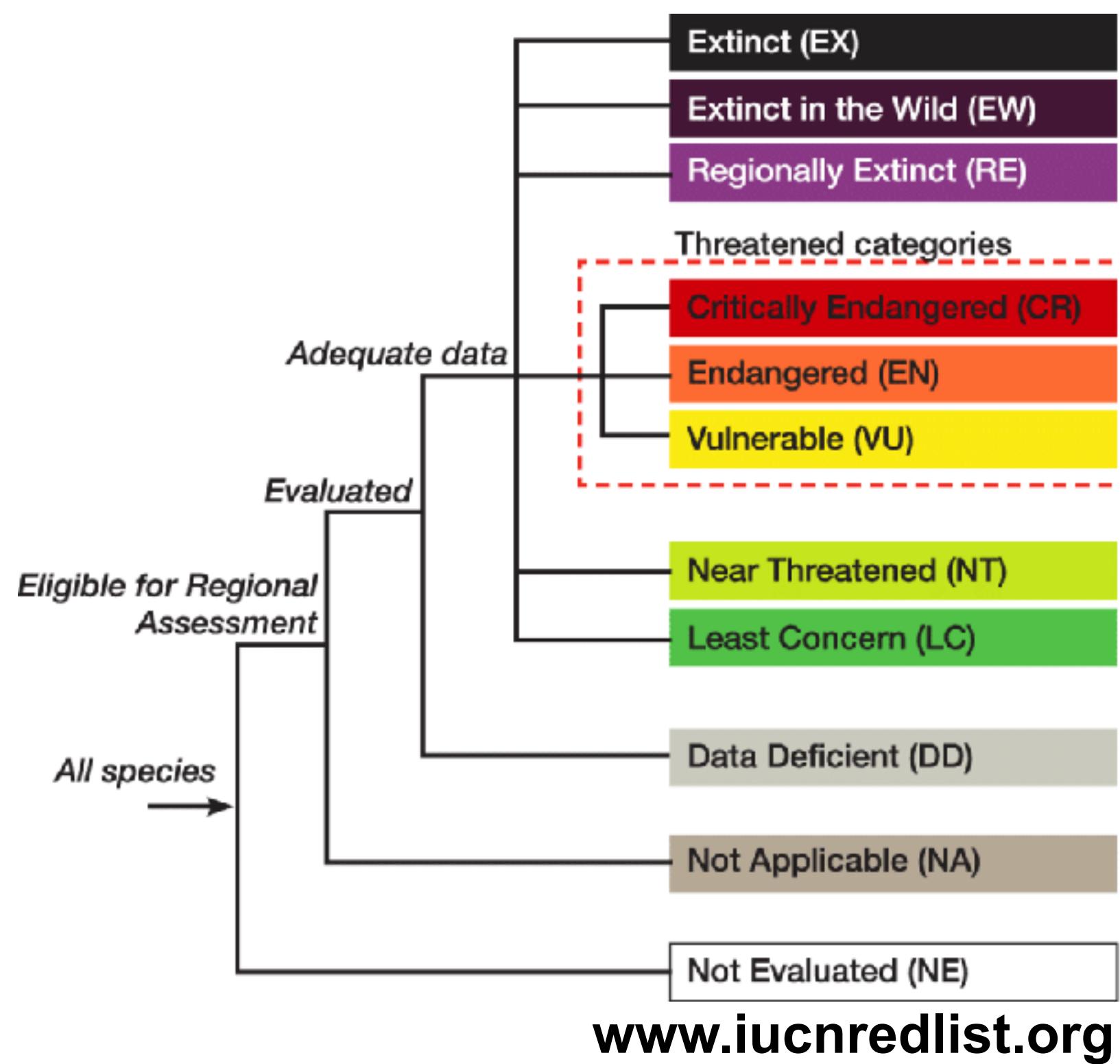


Modeling extinction risks

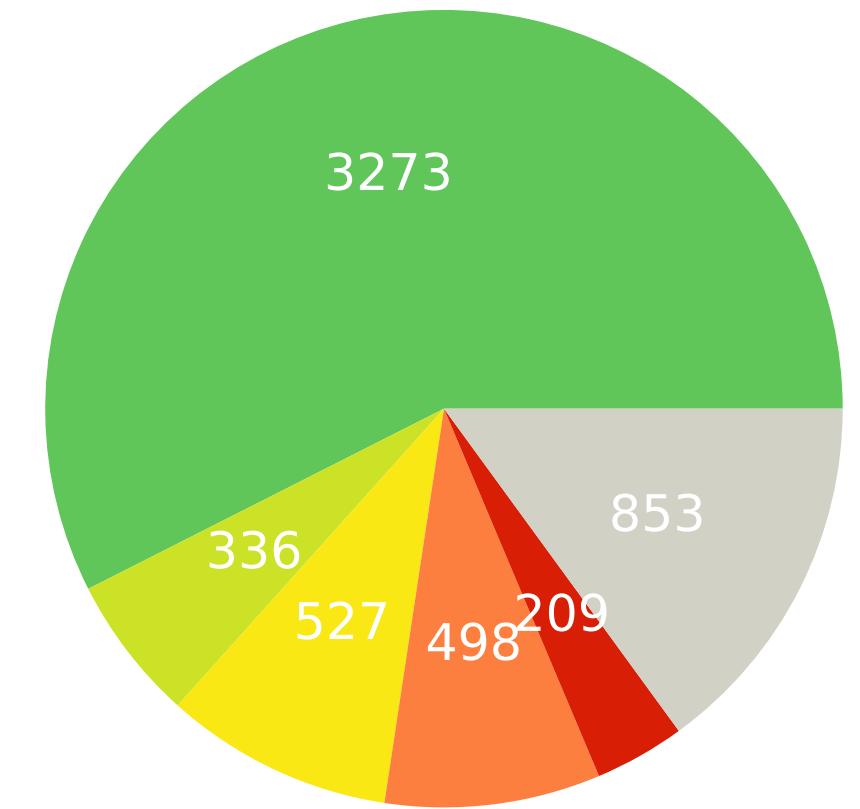
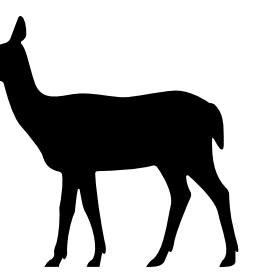




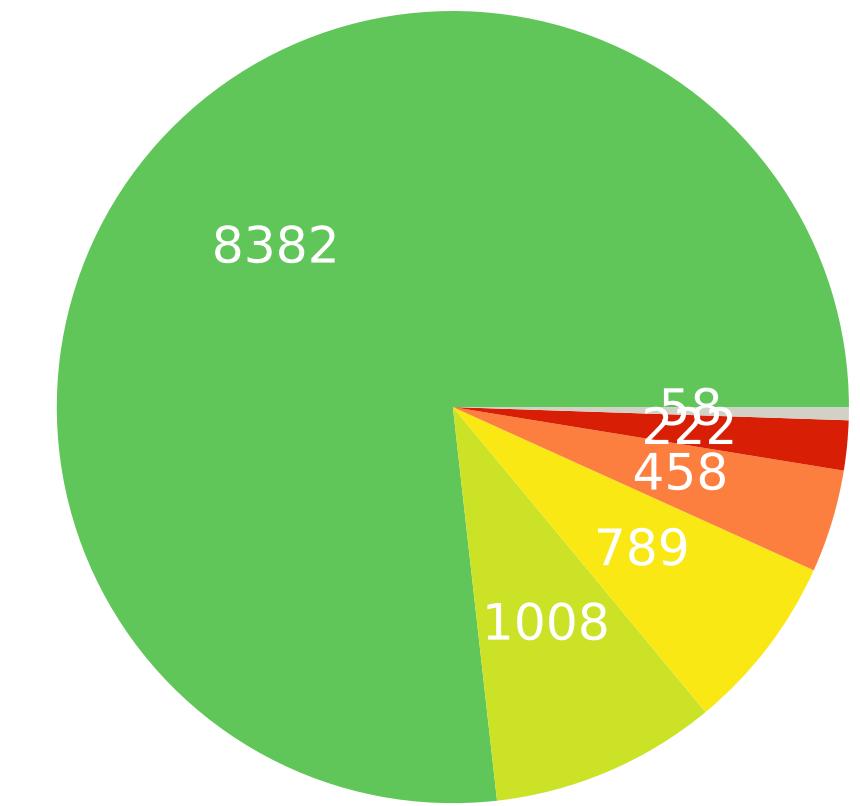
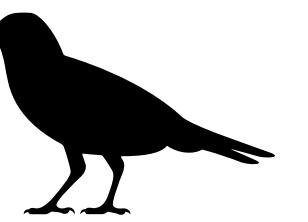
IUCN Redlist



Mammalia

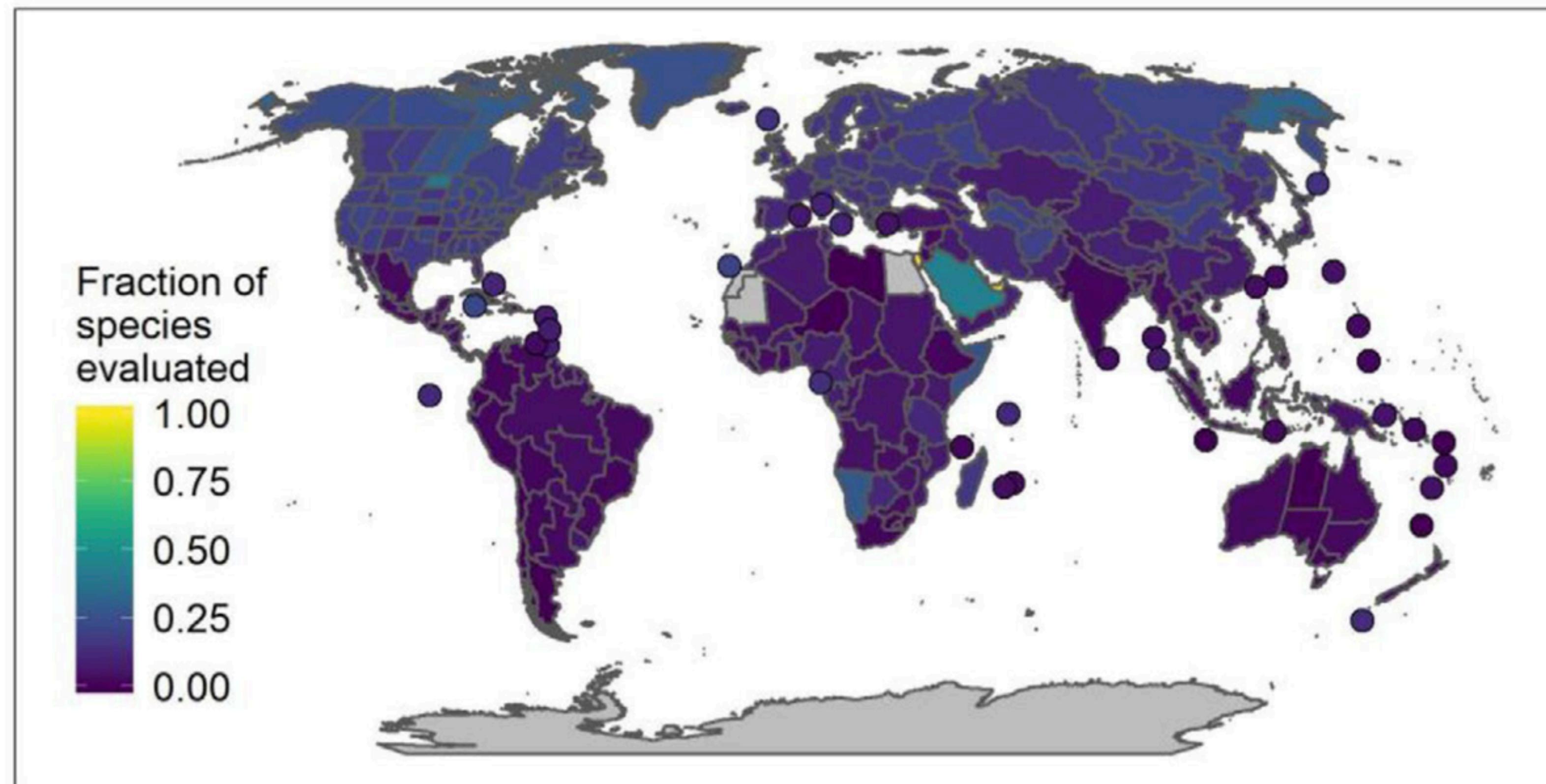


Aves



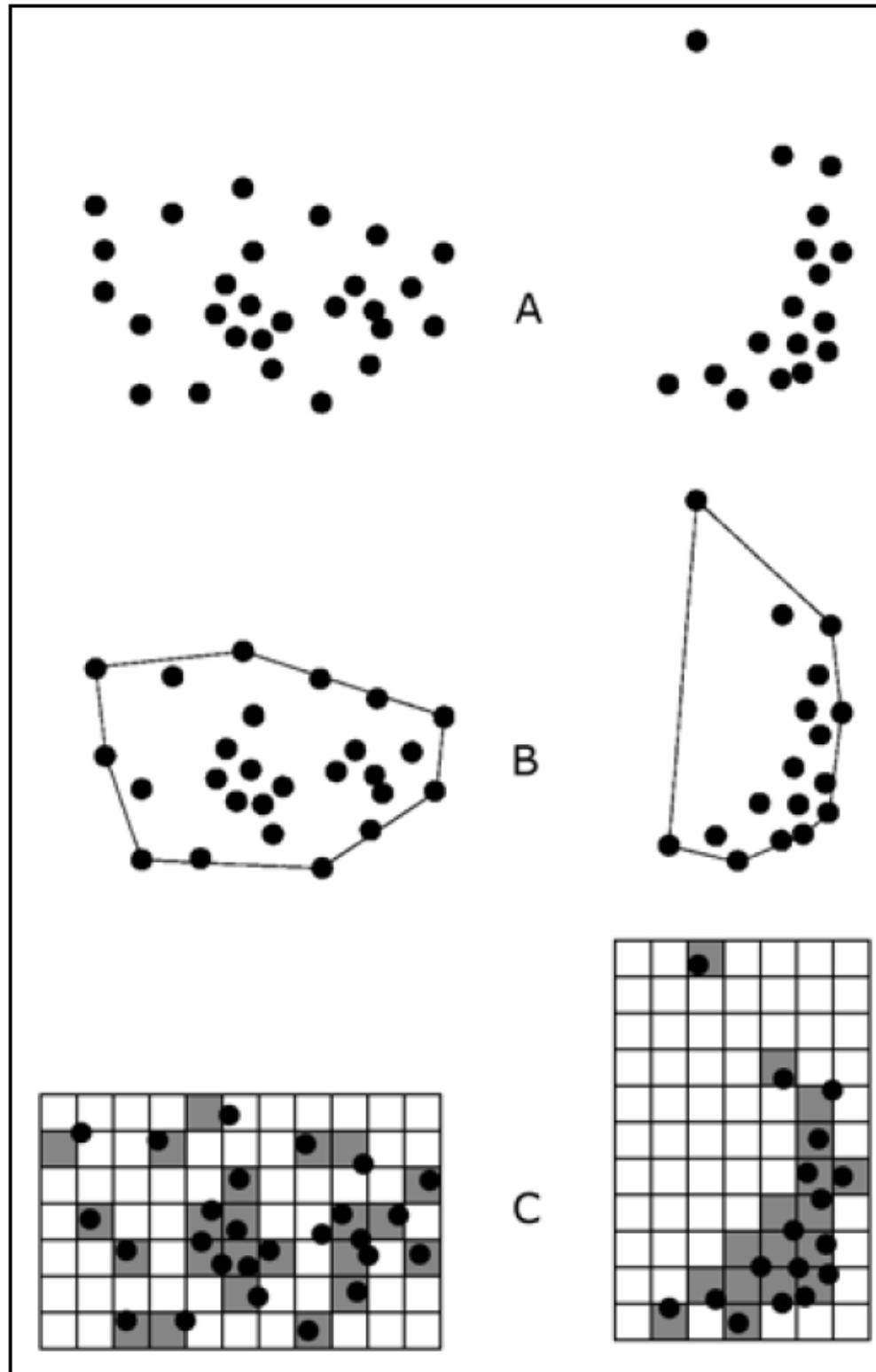
Much is still unknown

Fraction of orchid species assessed by IUCN



Zizka et al., 2020, Conservation Biology

Utility of distribution data

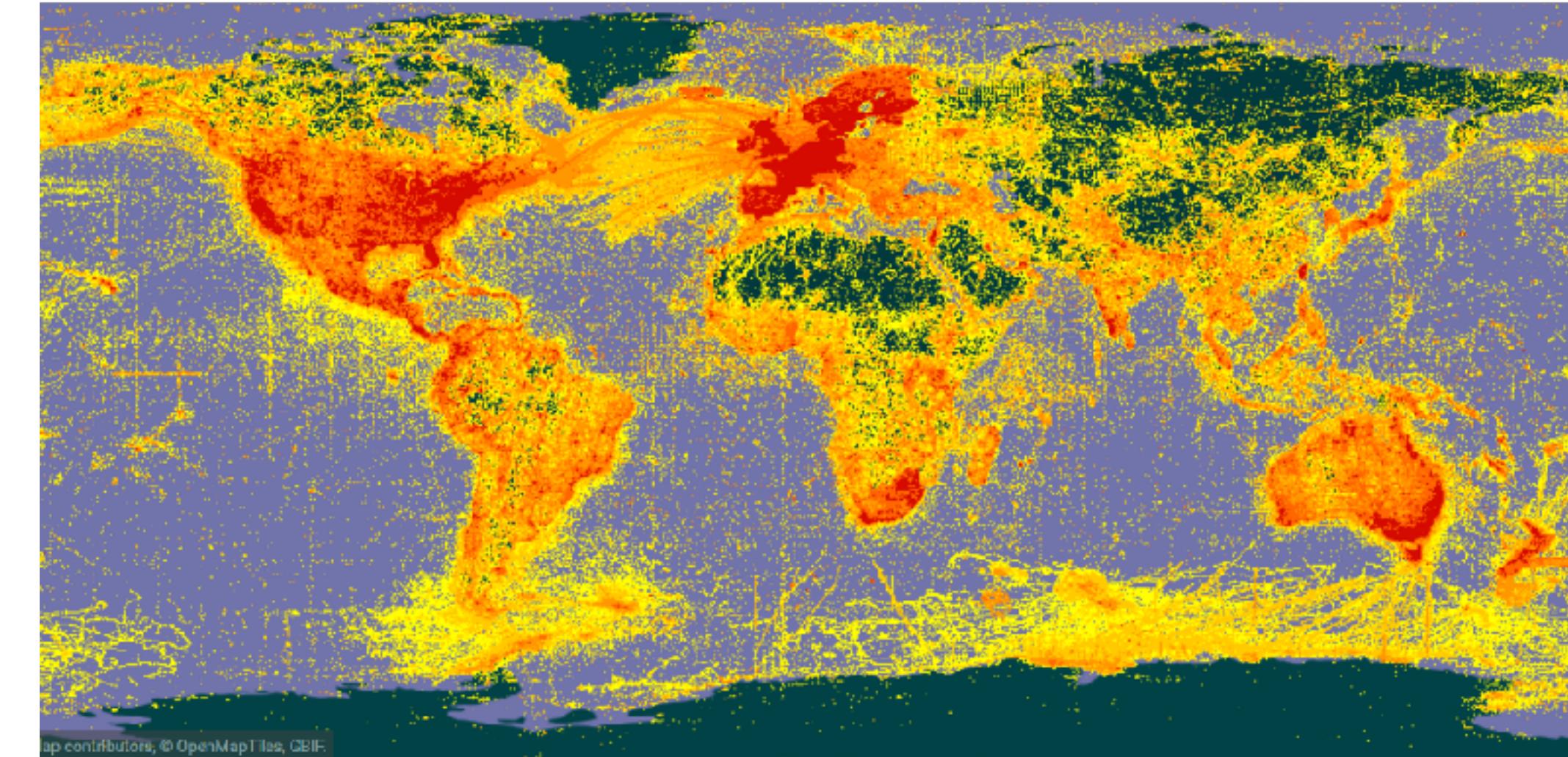


**IUCN Red List categories
and criteria**
www.iucnredlist.org

Occurrence
records

Extent of
occurrence

Area of
occupancy





Species occurrence data
from public databases



Longitude	Latitude
136.5329	-28.4025
150.8236	-24.8218
136.2989	-25.2995
133.7812	-25.9481
141.1054	-17.6605
141.7411	-31.0796
134.3736	-14.1506
133.6164	-22.2898
133.8822	-14.6768
141.817	-20.0843
135.409	-15.5381
140.8475	-17.4635
137.0116	-17.8959
143.7243	-20.1088
133.2699	-22.2836
136.5648	-25.7836
135.4576	-26.4805
133.6333	-23.8357
135.83	-16.8239
150.0949	-23.6582
141.1789	-17.295
139.53	-34.32
141.0037	-23.6464
135.8615	-28.0871
135.5817	-26.3667
134.3555	-14.177
133.4385	-25.7375
134.4207	-14.0479
142.4779	-20.0137
136.8653	-18.7861
135.3074	-26.5741

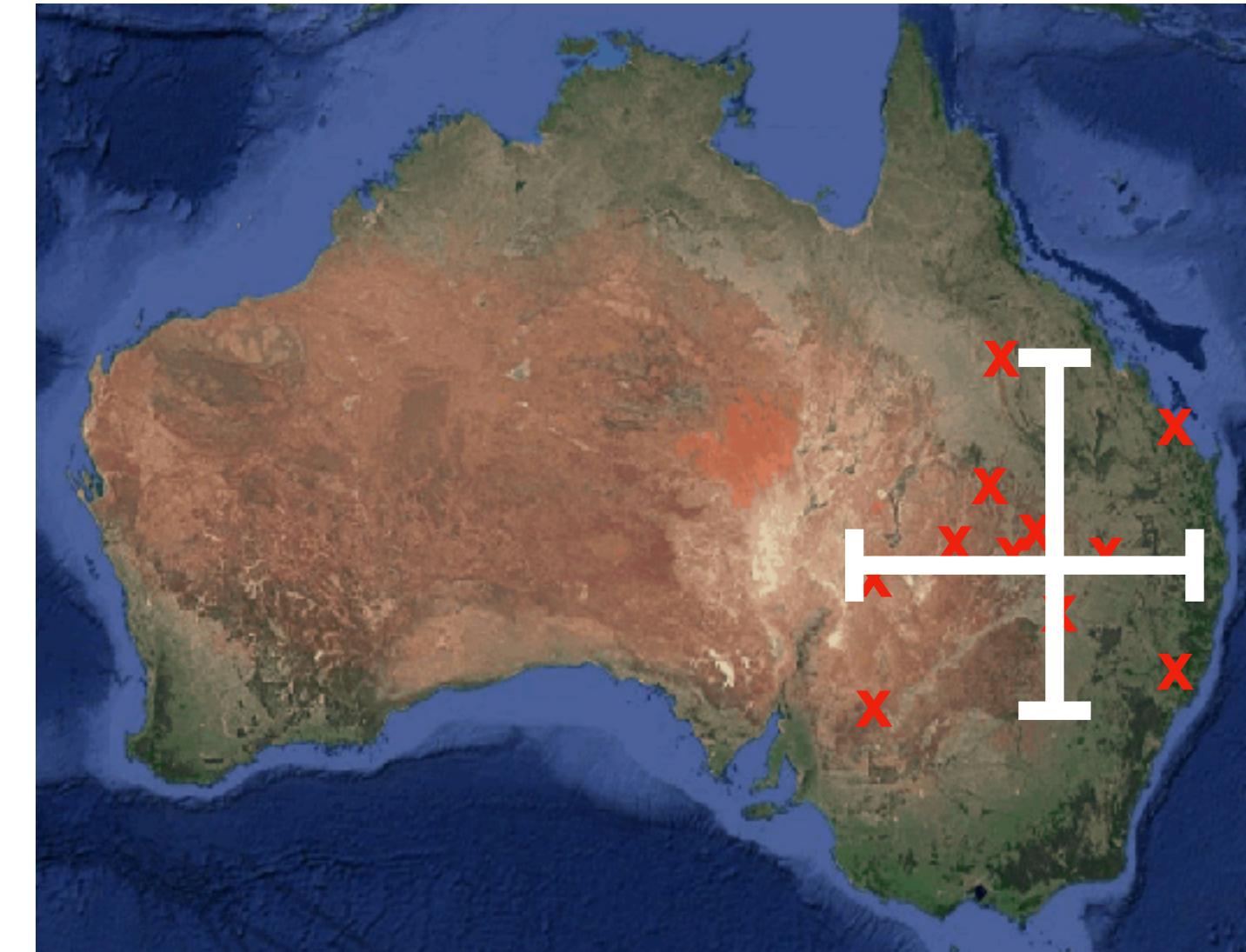


Geographic features

Species occurrence data
from public databases



Longitude	Latitude
136.5329	-28.4025
150.8236	-24.8218
136.2989	-25.2995
133.7812	-25.9481
141.1054	-17.6605
141.7411	-31.0796
134.3736	-14.1506
133.6164	-22.2898
133.8822	-14.6768
141.817	-20.0843
135.409	-15.5381
140.8475	-17.4635
137.0116	-17.8959
143.7243	-20.1088
133.2699	-22.2836
136.5648	-25.7836
135.4576	-26.4805
133.6333	-23.8357
135.83	-16.8239
150.0949	-23.6582
141.1789	-17.295
139.53	-34.32
141.0037	-23.6464
135.8615	-28.0871
135.5817	-26.3667
134.3555	-14.177
133.4385	-25.7375
134.4207	-14.0479
142.4779	-20.0137
136.8653	-18.7861
135.3074	-26.5741



Geographic features

- Total occurrences
- Longitudinal range
- Mean Longitude
- Latitudinal range
- Mean Latitude ...

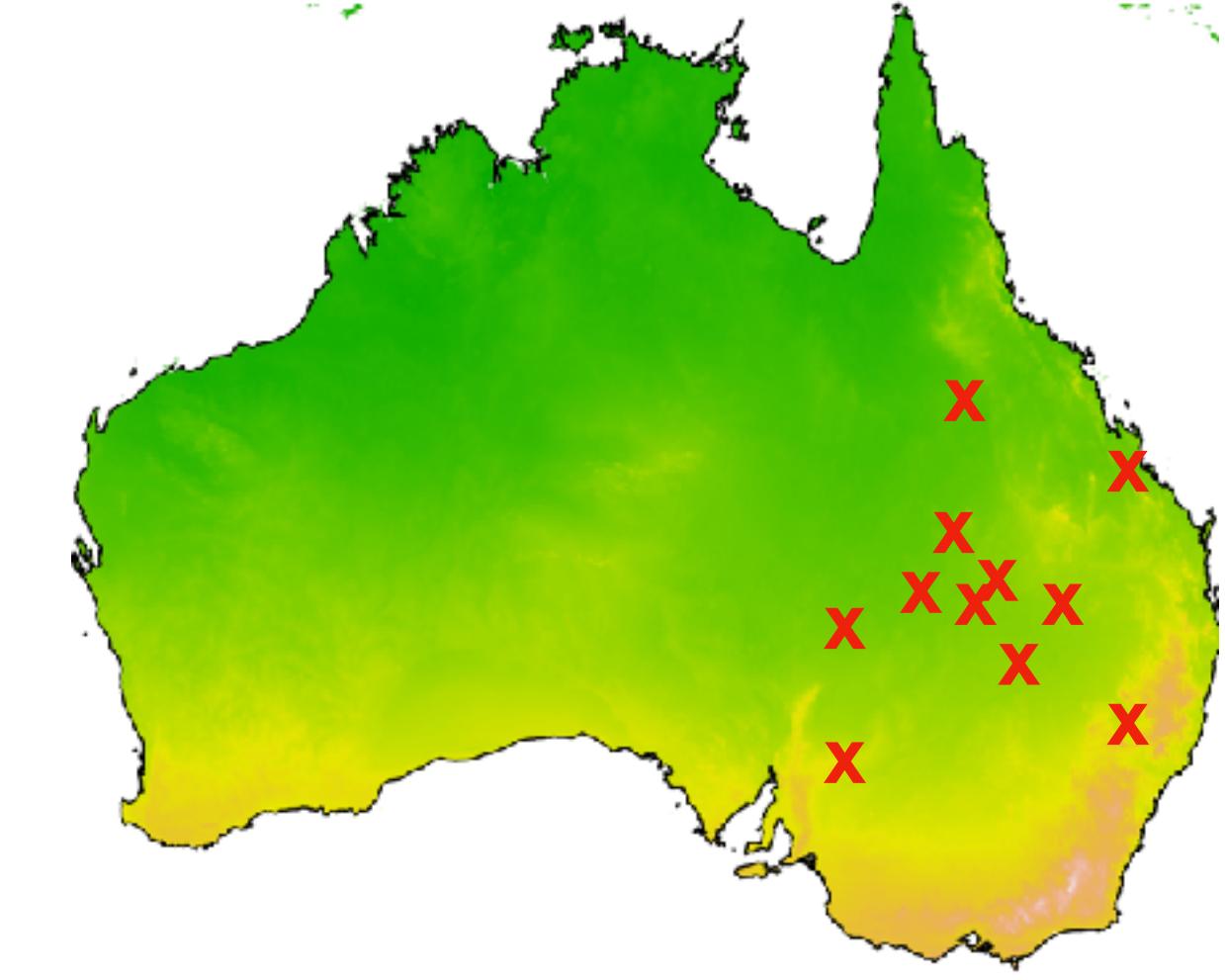
Species occurrence data
from public databases



Longitude	Latitude
136.5329	-28.4025
150.8236	-24.8218
136.2989	-25.2995
133.7812	-25.9481
141.1054	-17.6605
141.7411	-31.0796
134.3736	-14.1506
133.6164	-22.2898
133.8822	-14.6768
141.817	-20.0843
135.409	-15.5381
140.8475	-17.4635
137.0116	-17.8959
143.7243	-20.1088
133.2699	-22.2836
136.5648	-25.7836
135.4576	-26.4805
133.6333	-23.8357
135.83	-16.8239
150.0949	-23.6582
141.1789	-17.295
139.53	-34.32
141.0037	-23.6464
135.8615	-28.0871
135.5817	-26.3667
134.3555	-14.177
133.4385	-25.7375
134.4207	-14.0479
142.4779	-20.0137
136.8653	-18.7861
135.3074	-26.5741



Geographic features



Climatic features

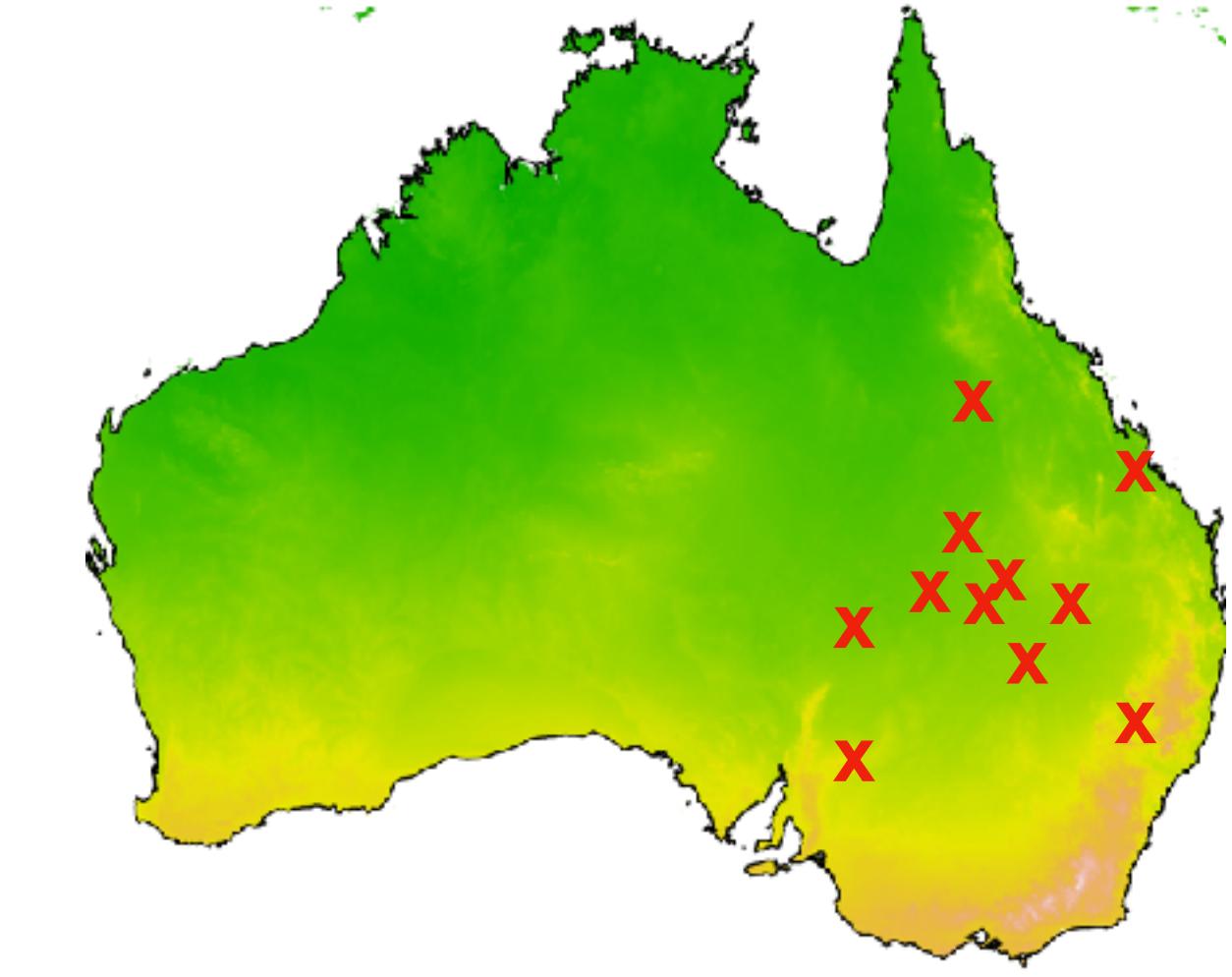
Species occurrence data
from public databases



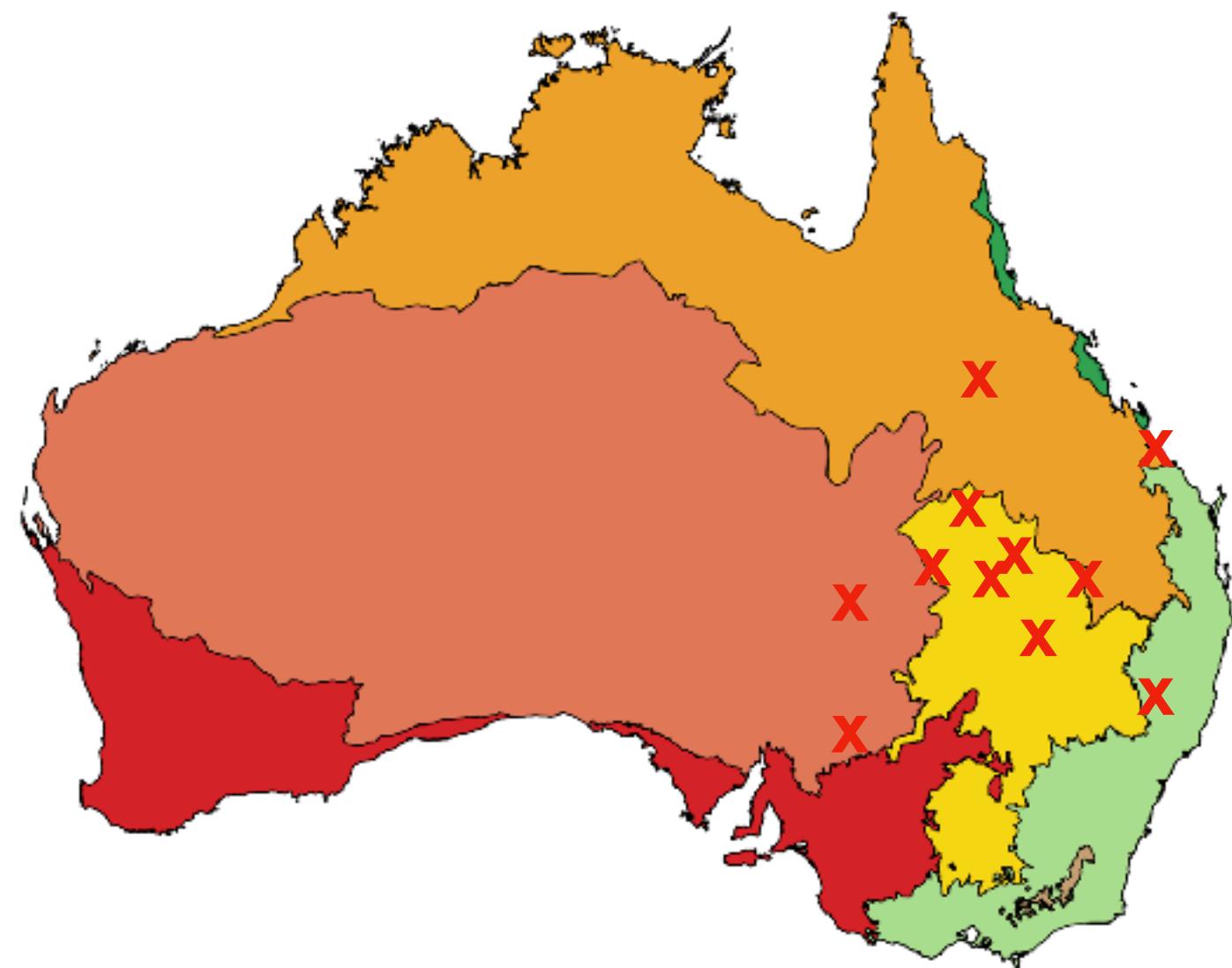
Longitude	Latitude
136.5329	-28.4025
150.8236	-24.8218
136.2989	-25.2995
133.7812	-25.9481
141.1054	-17.6605
141.7411	-31.0796
134.3736	-14.1506
133.6164	-22.2898
133.8822	-14.6768
141.817	-20.0843
135.409	-15.5381
140.8475	-17.4635
137.0116	-17.8959
143.7243	-20.1088
133.2699	-22.2836
136.5648	-25.7836
135.4576	-26.4805
133.6333	-23.8357
135.83	-16.8239
150.0949	-23.6582
141.1789	-17.295
139.53	-34.32
141.0037	-23.6464
135.8615	-28.0871
135.5817	-26.3667
134.3555	-14.177
133.4385	-25.7375
134.4207	-14.0479
142.4779	-20.0137
136.8653	-18.7861
135.3074	-26.5741



Geographic features



Climatic features



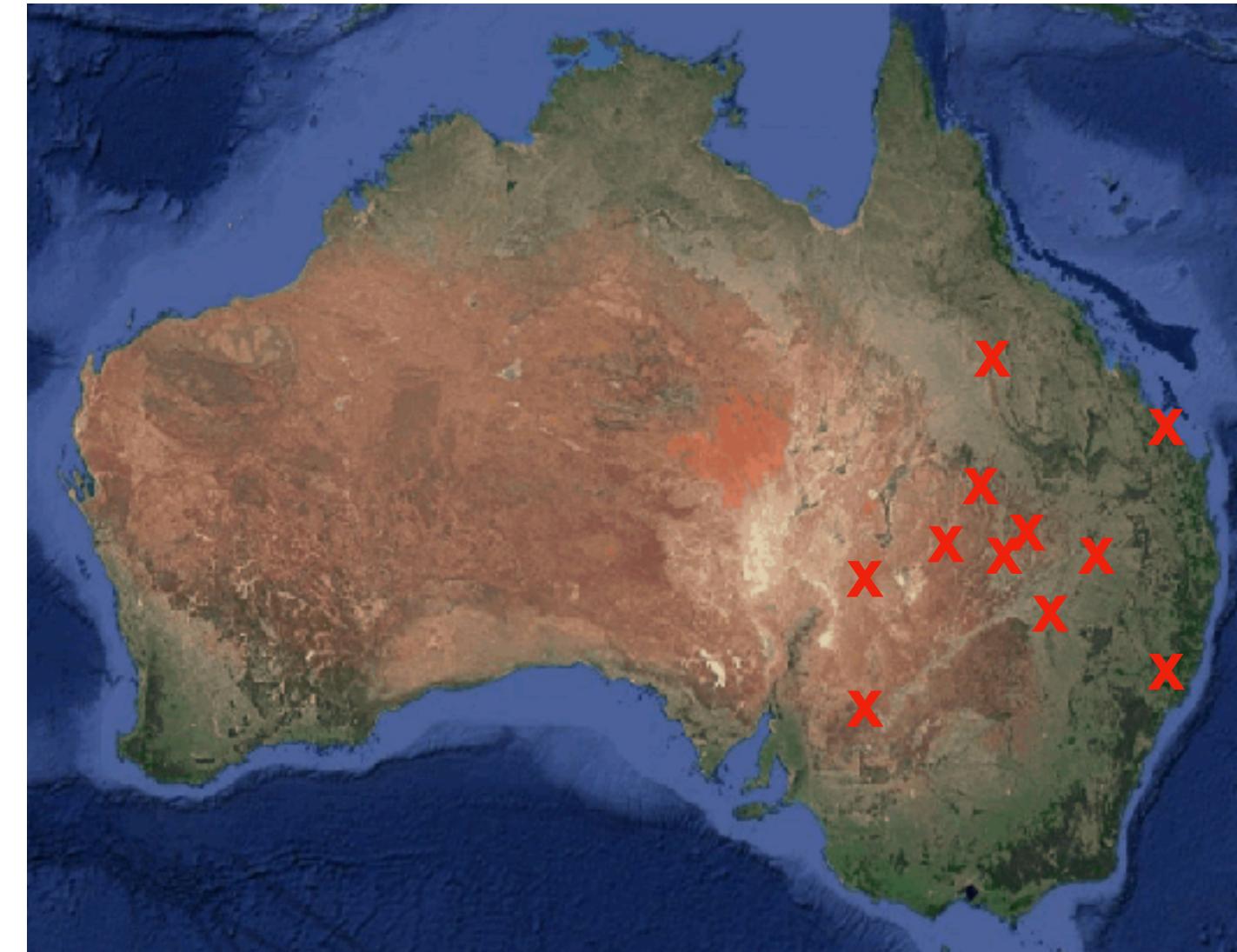
Biome features

Species occurrence data
from public databases

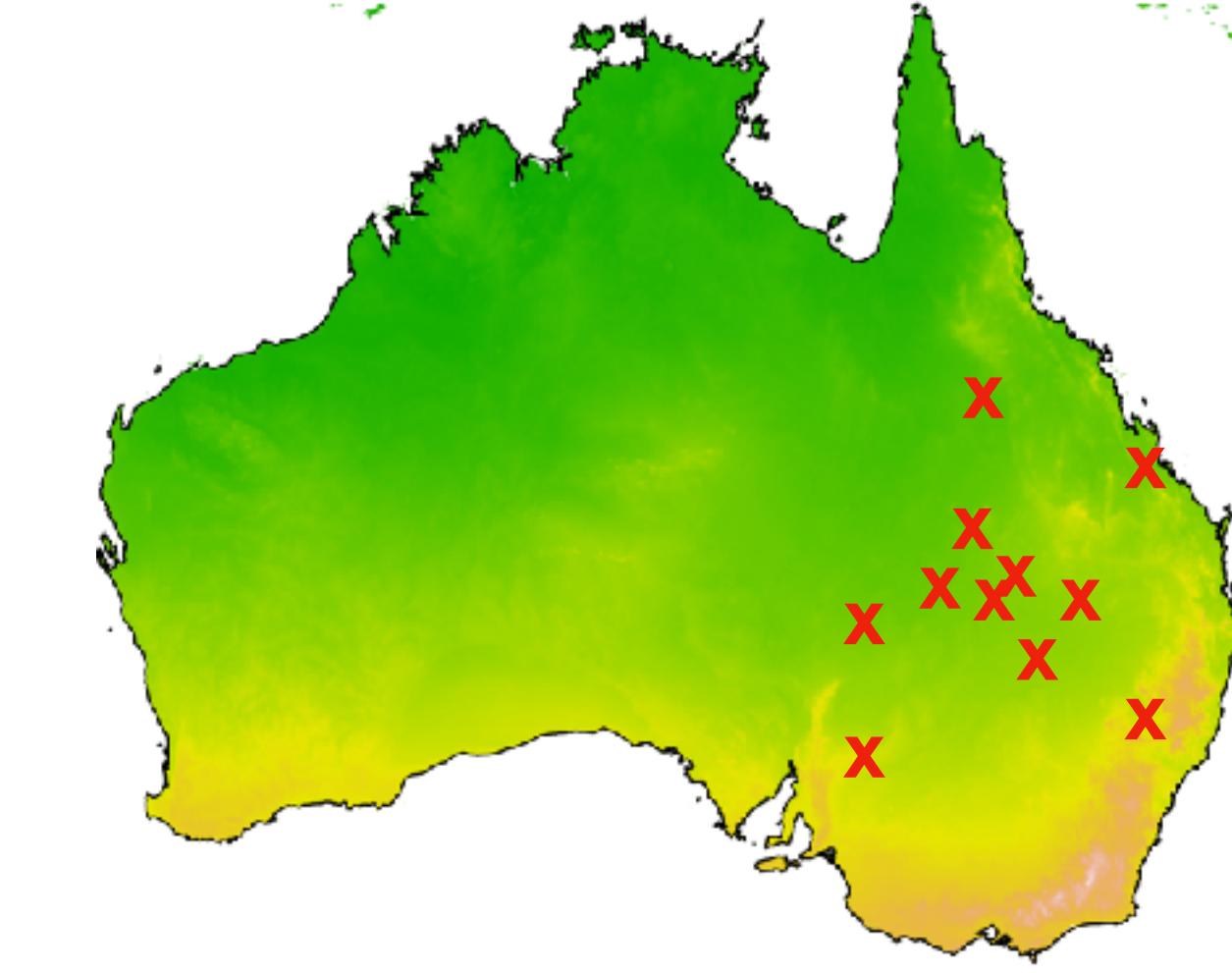


Longitude	Latitude
136.5329	-28.4025
150.8236	-24.8218
136.2989	-25.2995
133.7812	-25.9481
141.1054	-17.6605
141.7411	-31.0796
134.3736	-14.1506
133.6164	-22.2898
133.8822	-14.6768
141.817	-20.0843
135.409	-15.5381
140.8475	-17.4635
137.0116	-17.8959
143.7243	-20.1088
133.2699	-22.2836
136.5648	-25.7836
135.4576	-26.4805
133.6333	-23.8357
135.83	-16.8239
150.0949	-23.6582
141.1789	-17.295
139.53	-34.32
141.0037	-23.6464
135.8615	-28.0871
135.5817	-26.3667
134.3555	-14.177
133.4385	-25.7375
134.4207	-14.0479
142.4779	-20.0137
136.8653	-18.7861
135.3074	-26.5741

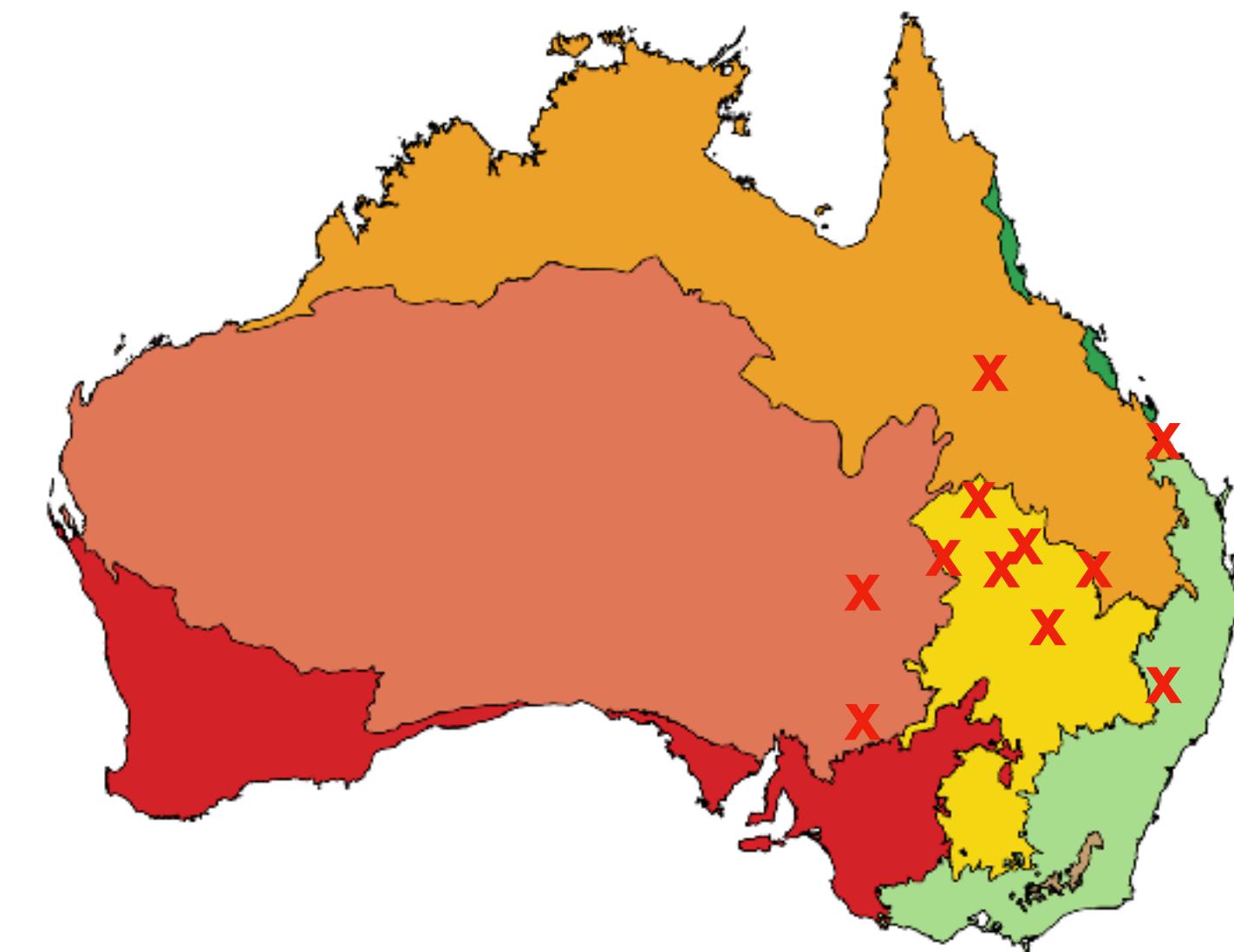
- Total occurrences
- Longitudinal range
- Mean Longitude
- Latitudinal range
- Mean Latitude ...
- Median temperature
- Range of temperature
- Median precipitation ...
- Presence in biome 1
- Presence in biome 2 ...
- Fraction of records with high human footprint
- Fraction of records with medium human footprint
- ...



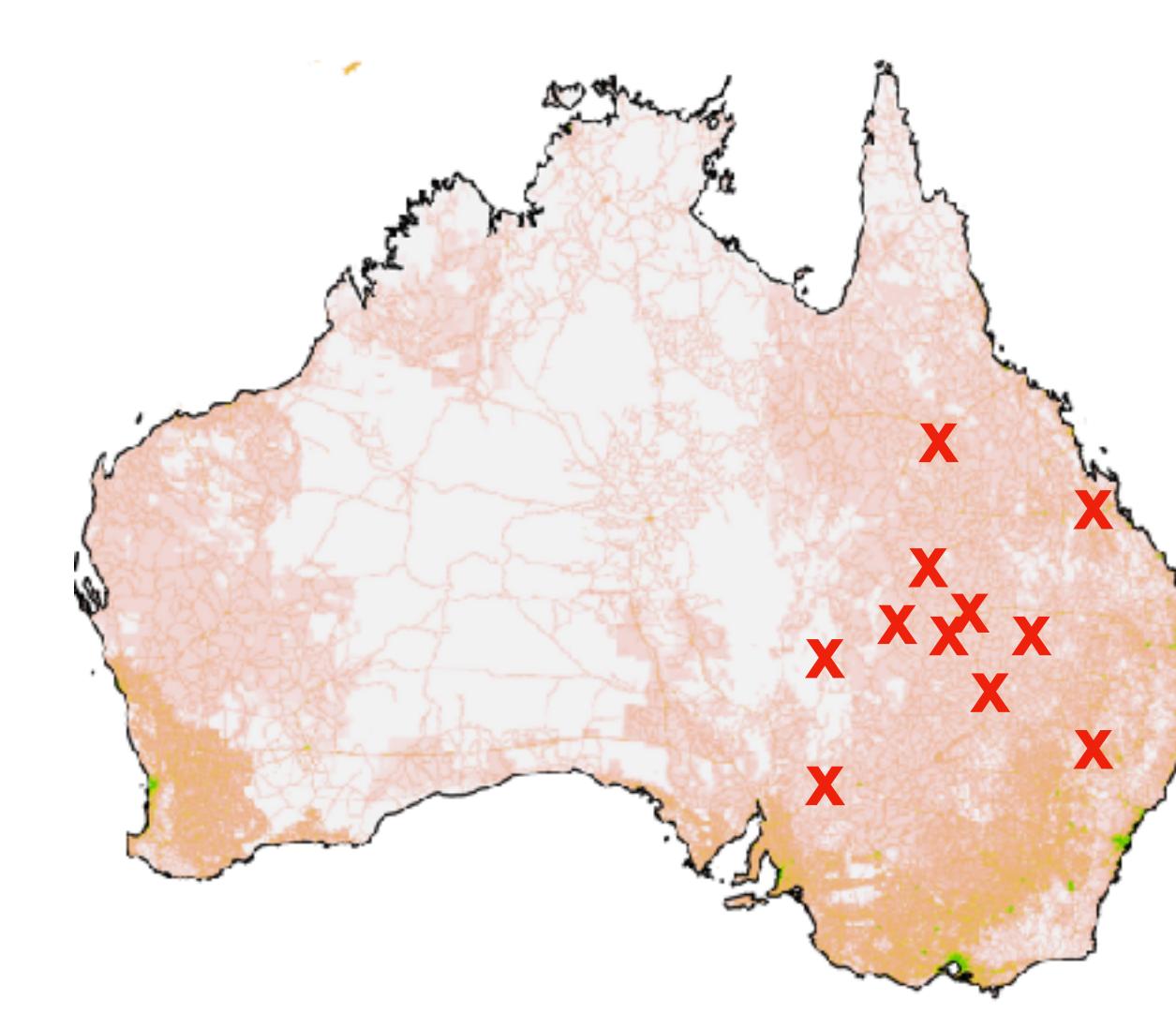
Geographic features



Climatic features



Biome features

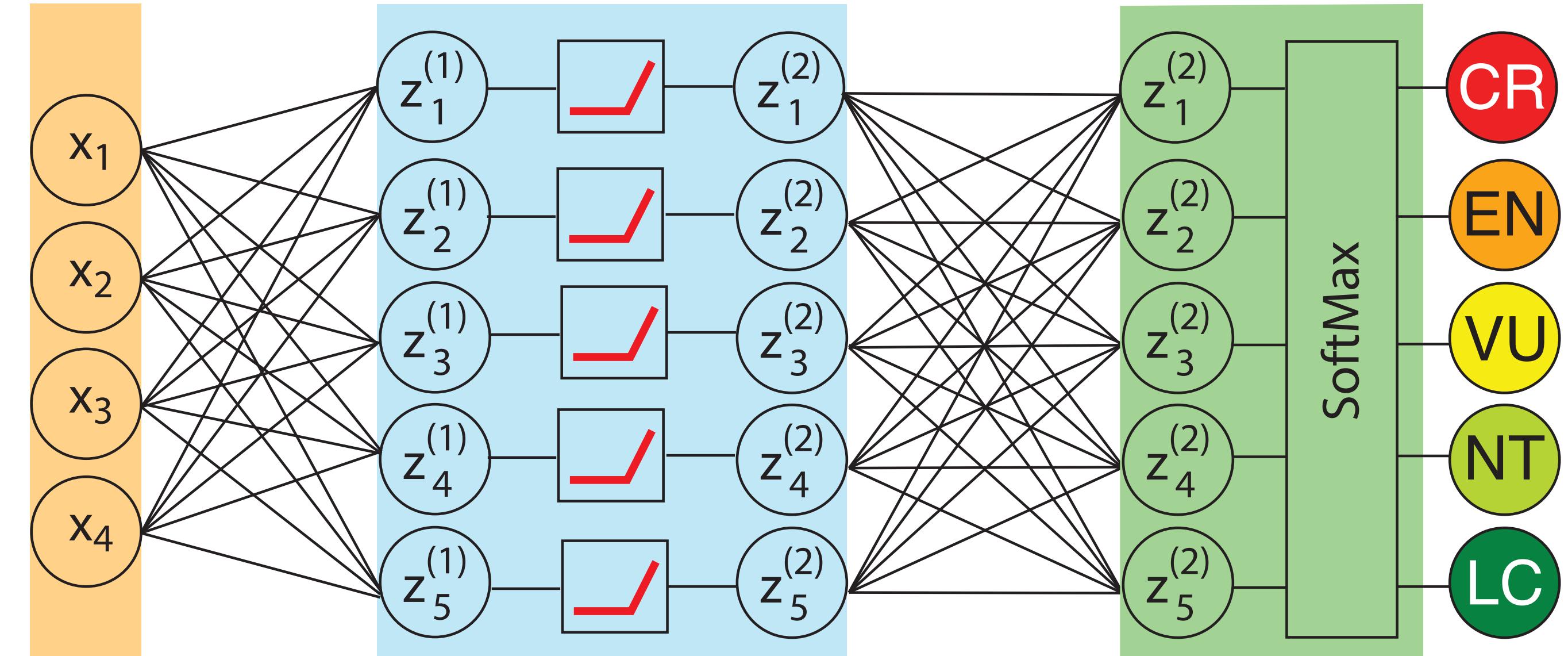


Human footprint features

Threat status classification



Geographic (n. occs, EOO, AOO)
Climatic (bioclim variables)
Biome (presence/absence)
Anthropogenic (human footprint index)



Diversity and Distributions
A Journal of Conservation Biogeography
Open Access

METHOD | Open Access | CC BY

Model implemented in :
IUCNN – Deep learning approaches to approximate species' extinction risk

Alexander Zizka, Tobias Andermann, Daniele Silvestro

First published: 16 December 2021 | <https://doi.org/10.1111/ddi.13450> | Citations: 1

```

# Example for classification model

architecture = []

# Input layer
architecture.append(tf.keras.layers.Flatten(input_shape=[4]))

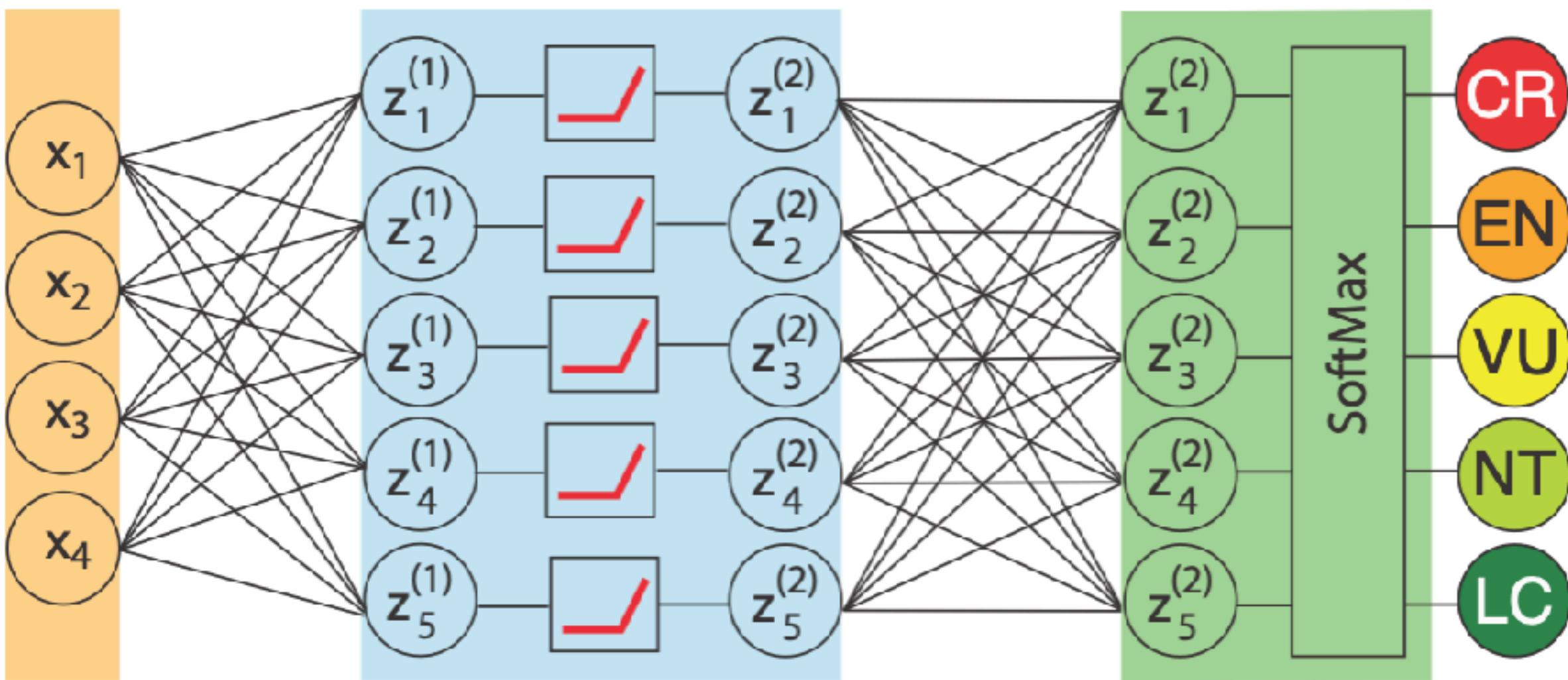
# 1st hidden layer
architecture.append(tf.keras.layers.Dense(5, activation='relu', use_bias=False))

# Output layer
architecture.append(tf.keras.layers.Dense(5, activation='softmax', use_bias=False))

# Compile the model
model = tf.keras.Sequential(architecture)
model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])

# Get overview of model architecture
model.summary()

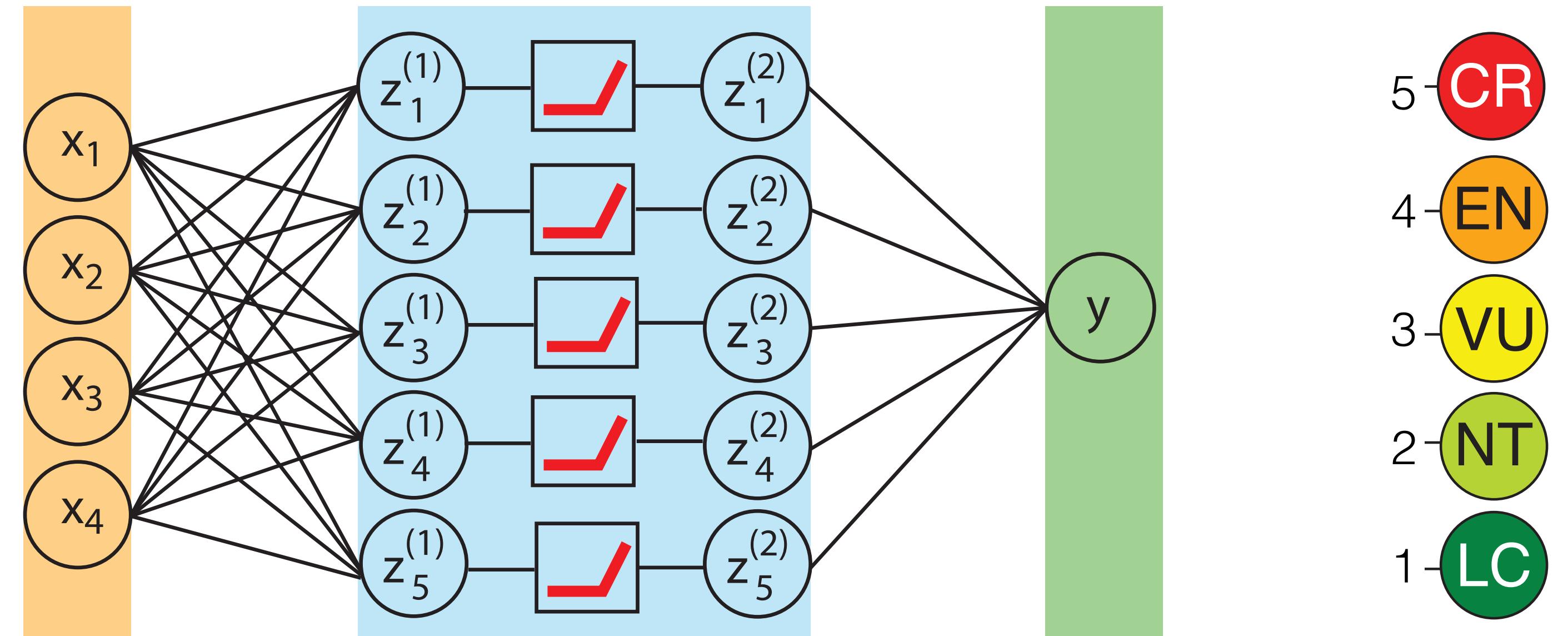
```



Threat status regression model



Geographic (n. occs, EOO, AOO)
Climatic (bioclim variables)
Biome (presence/absence)
Anthropogenic (human footprint index)



Diversity and Distributions
Open Access

A Journal of
Conservation
Biogeography

METHOD | Open Access | CC BY

Model implemented in :
IUCNN – Deep learning approaches to approximate species' extinction risk

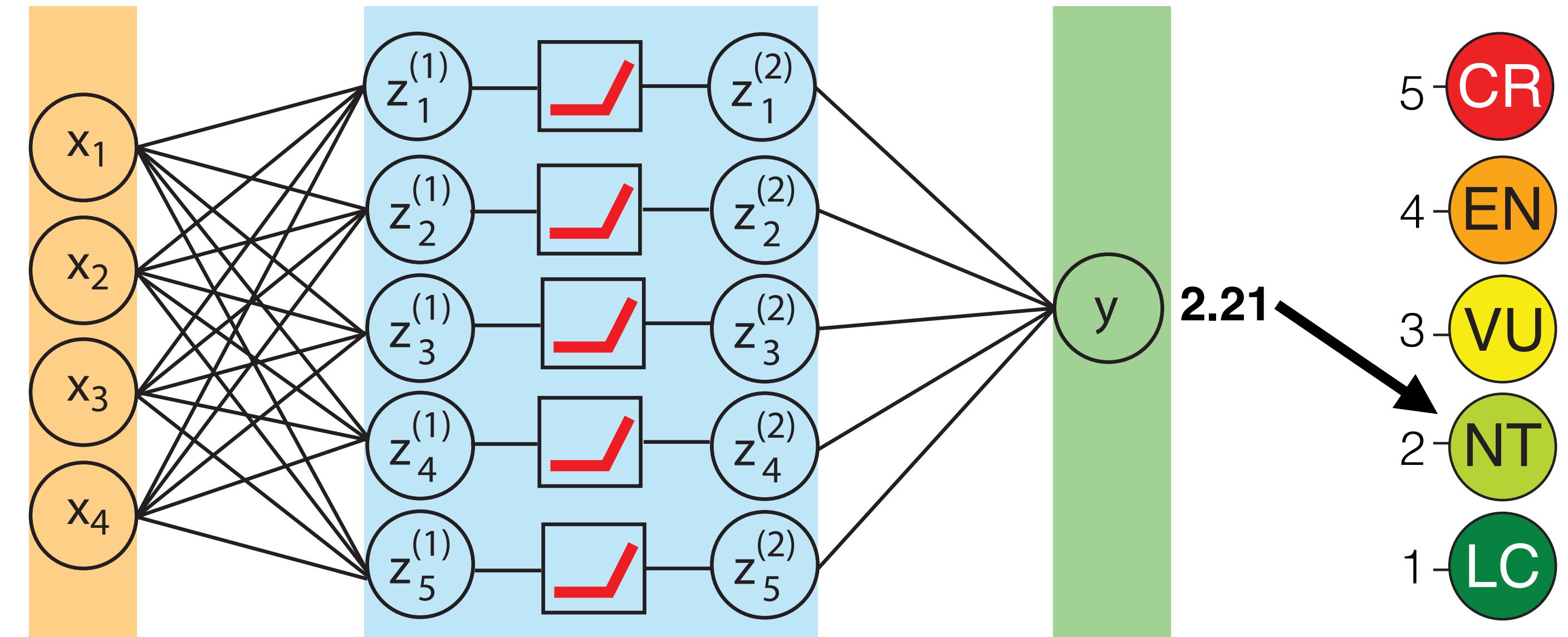
Alexander Zizka, Tobias Andermann, Daniele Silvestro

First published: 16 December 2021 | <https://doi.org/10.1111/ddi.13450> | Citations: 1

Threat status regression model



Geographic (n. occs, EOO, AOO)
Climatic (bioclim variables)
Biome (presence/absence)
Anthropogenic (human footprint index)



Diversity and Distributions
A Journal of Conservation Biogeography
Open Access

METHOD | Open Access | CC BY
IUCNN – Deep learning approaches to approximate species' extinction risk

Alexander Zizka, Tobias Andermann, Daniele Silvestro

First published: 16 December 2021 | <https://doi.org/10.1111/ddi.13450> | Citations: 1

```

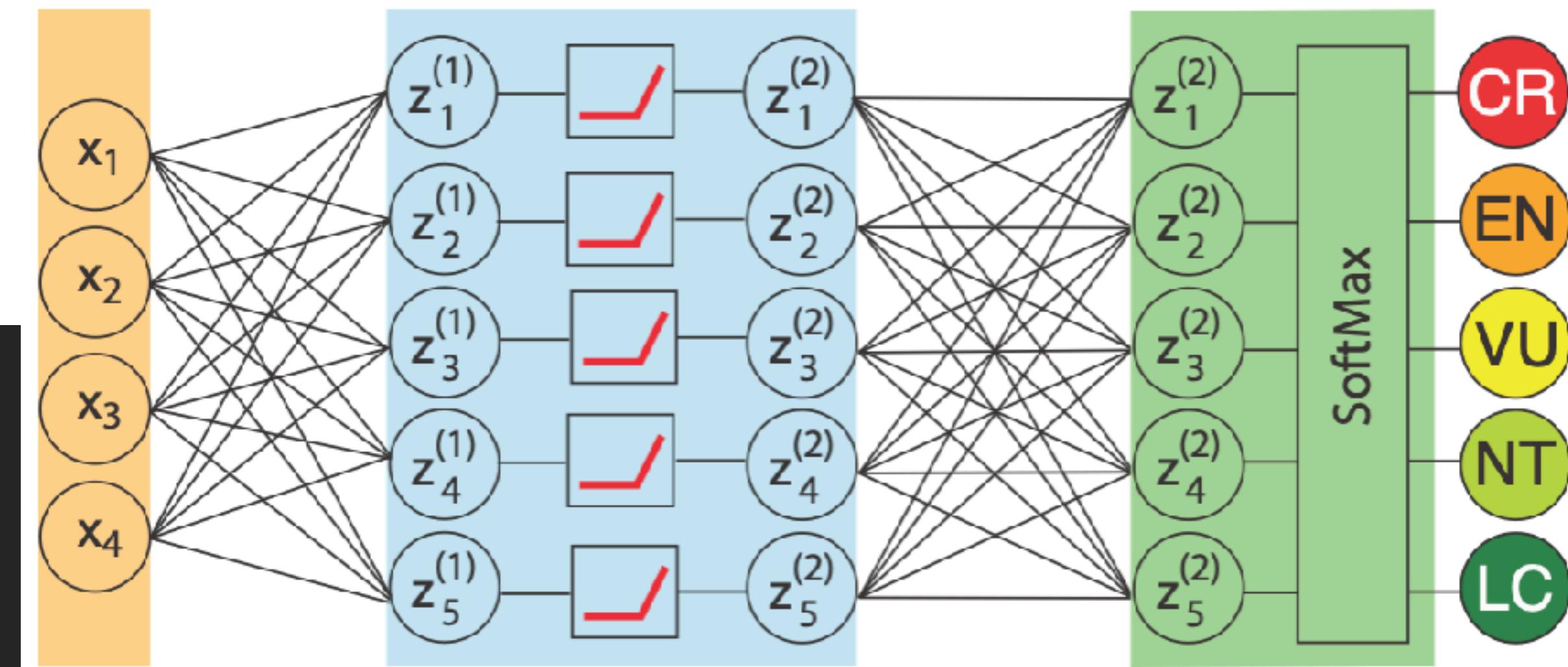
# Example for classification model
architecture = []

# Input layer
architecture.append(tf.keras.layers.Flatten(input_shape=[4]))
# 1st hidden layer
architecture.append(tf.keras.layers.Dense(5, activation='relu',use_bias=False))
# Output layer
architecture.append(tf.keras.layers.Dense(5, activation='softmax',use_bias=False))

# Compile the model
model = tf.keras.Sequential(architecture)
model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])

# Get overview of model architecture
model.summary()

```



```

# Example for regression model

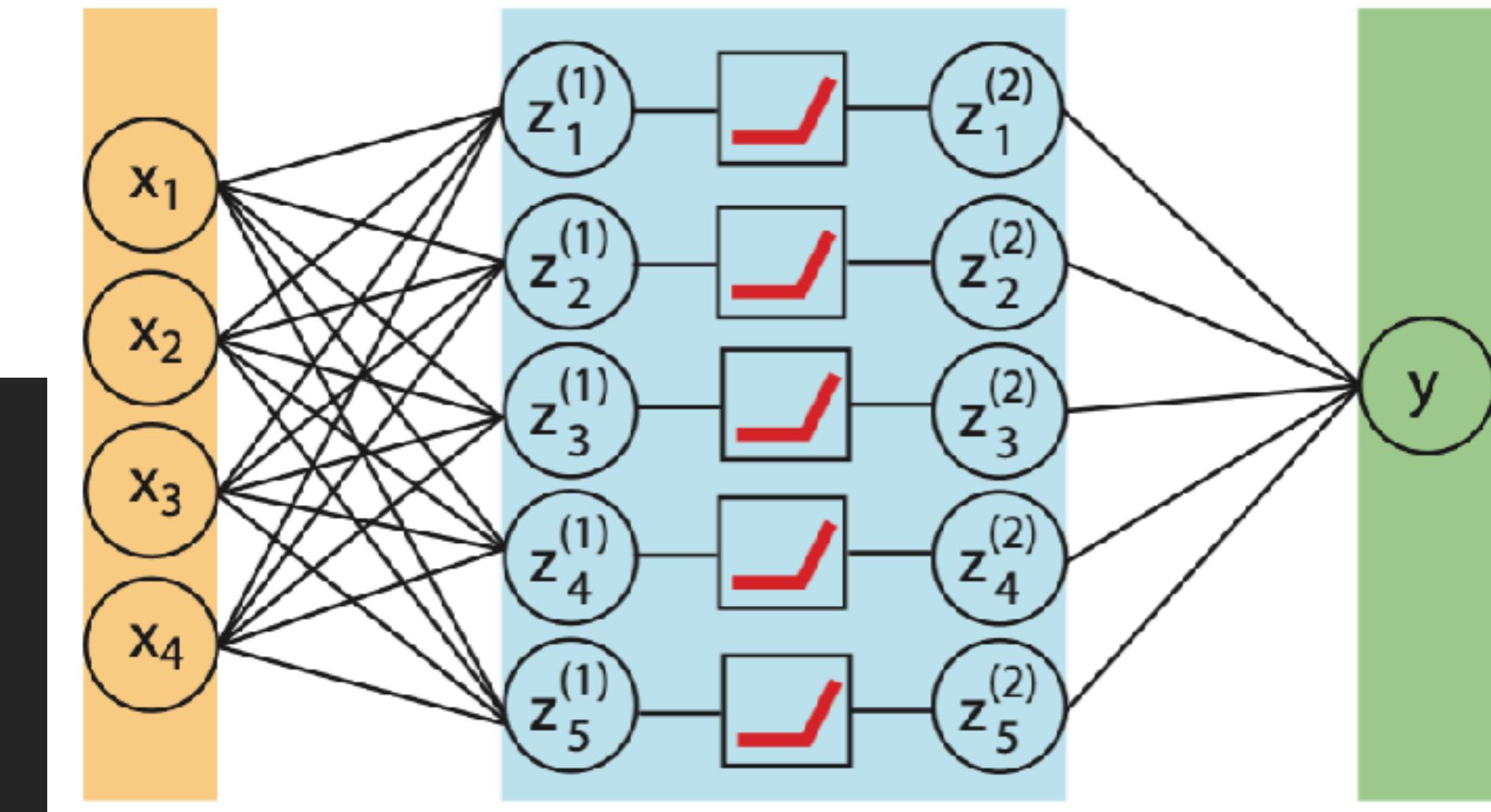
architecture = []

# Input layer
architecture.append(tf.keras.layers.Flatten(input_shape=[4]))
# 1st hidden layer
architecture.append(tf.keras.layers.Dense(5, activation='relu',use_bias=False))
# Output layer
architecture.append(tf.keras.layers.Dense(1, activation='softplus',use_bias=False))

# Compile the model
model = tf.keras.Sequential(architecture)
model.compile(loss='mae',
              optimizer='adam', metrics=['mae'])

# Get overview of model architecture
model.summary()

```



- 5 - CR
- 4 - EN
- 3 - VU
- 2 - NT
- 1 - LC

```

# Example for regression model

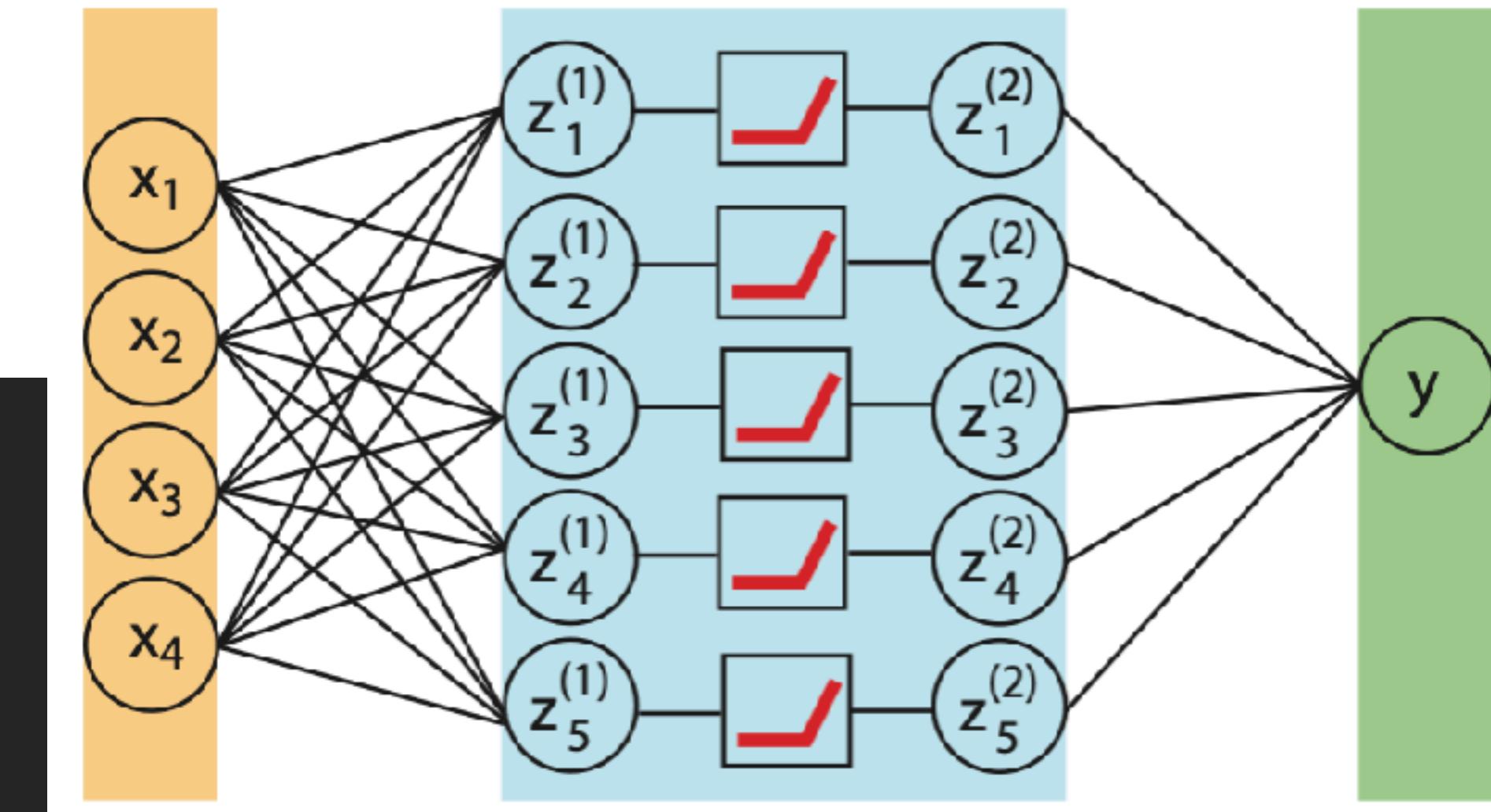
architecture = []

# Input layer
architecture.append(tf.keras.layers.Flatten(input_shape=[4]))
# 1st hidden layer
architecture.append(tf.keras.layers.Dense(5, activation='relu', use_bias=False))
# Output layer
architecture.append(tf.keras.layers.Dense(1, activation='softplus', use_bias=False))

# Compile the model
model = tf.keras.Sequential(architecture)
model.compile(loss='mae',
              optimizer='adam', metrics=['mae'])

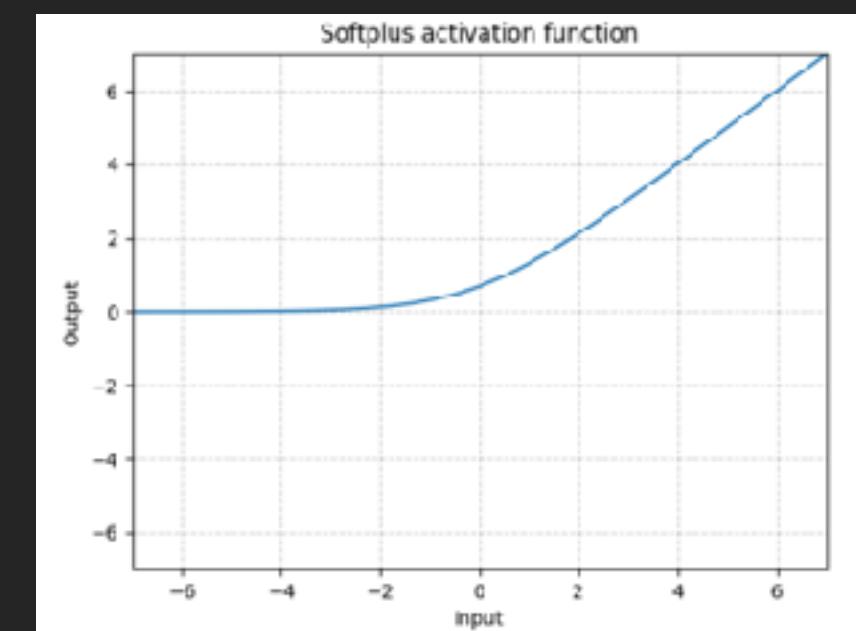
# Get overview of model architecture
model.summary()

```



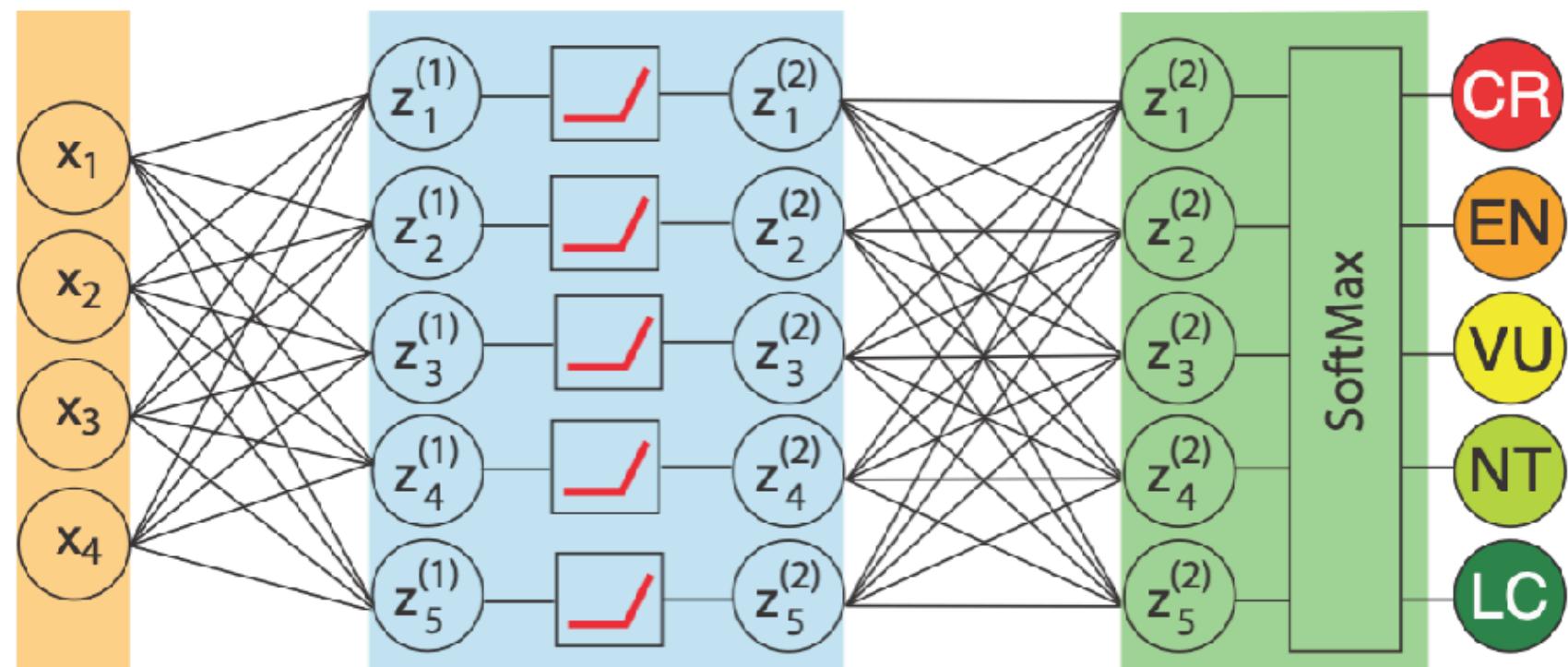
- 5 - CR
- 4 - EN
- 3 - VU
- 2 - NT
- 1 - LC

Softplus activation function



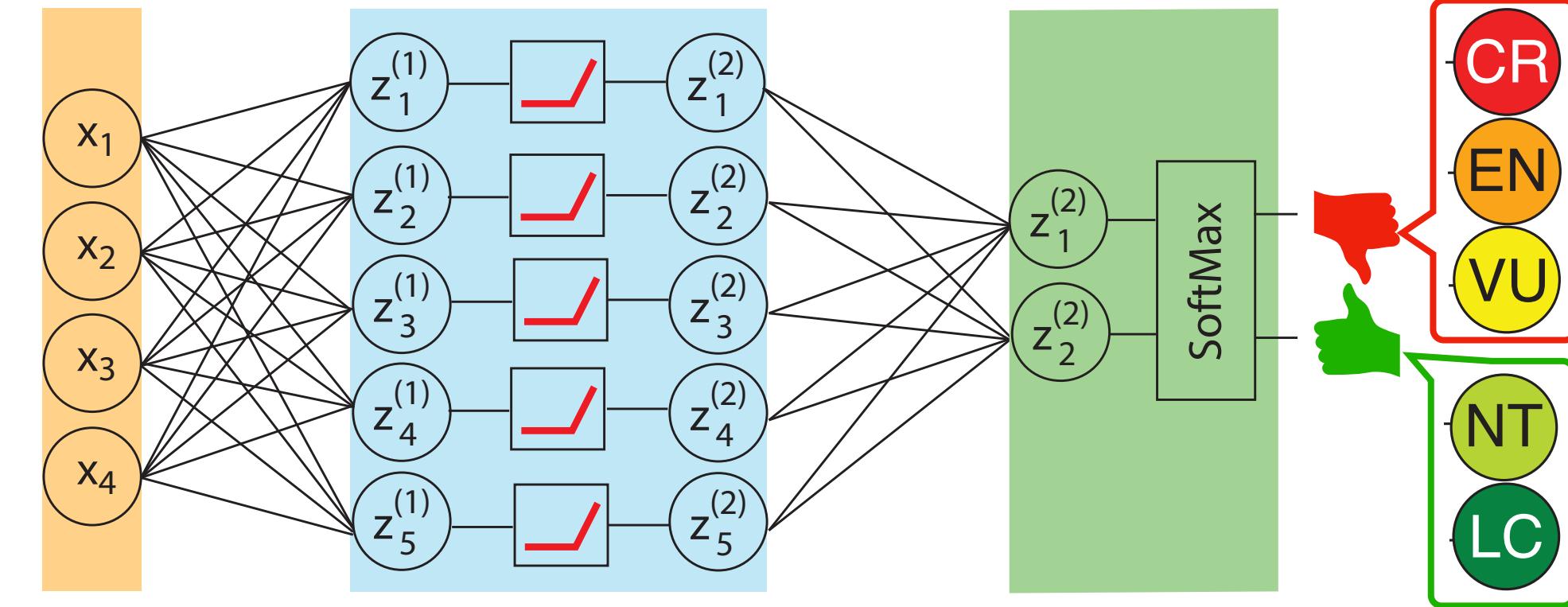
Model performance

5-class model



64% test-set accuracy

2-class model: Threatened or not threatened



84% test-set accuracy

Classification

		nn-class				
		LC	NT	VU	EN	CR
LC	LC	328	0	6	43	6
	NT	43	0	1	13	1
	VU	57	0	5	61	1
	EN	45	0	8	157	17
	CR	10	0	2	38	43

Regression

		nn-reg				
		LC	NT	VU	EN	CR
LC	LC	265	53	46	16	3
	NT	20	19	14	5	0
	VU	30	18	52	23	1
	EN	22	16	70	116	3
	CR	5	4	23	47	14

Classification

		nn-class	
Possibly Threatened	Threatened	Not Threatened	Possibly Threatened
		Not Threatened	Possibly Threatened
Not Threatened	358	83	
Possibly Threatened	85	359	

Regression

		nn-reg	
Possibly Threatened	Threatened	Not Threatened	Possibly Threatened
		Not Threatened	Possibly Threatened
Not Threatened	349	92	
Possibly Threatened	83	361	

IUCNN R-package

repo status Active DOI 10.5281/zenodo.5510580

IUCNN

Batch estimation of species' IUCN Red List threat status using neural networks.

Installation

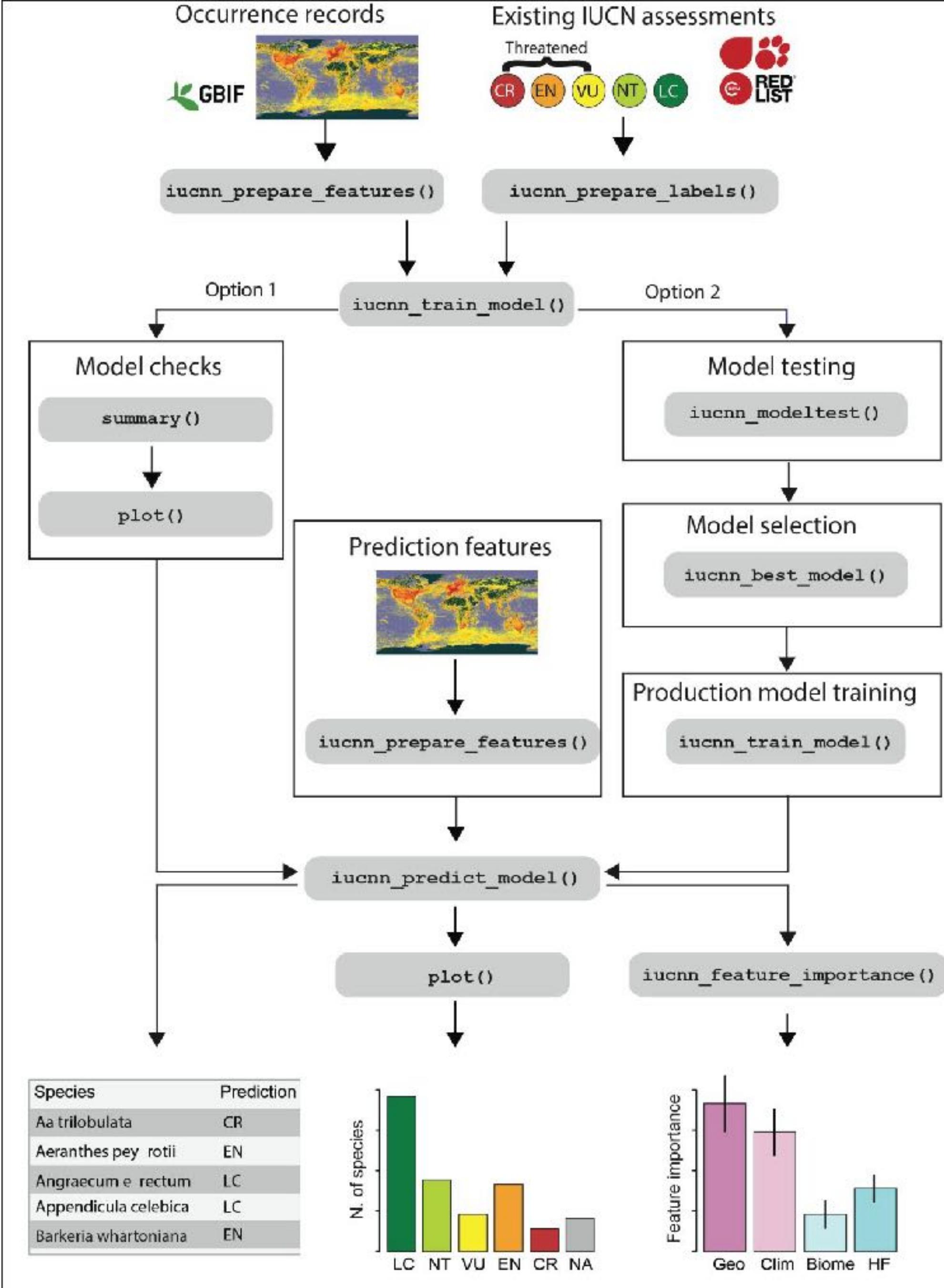
1. Install IUCNN directly from Github using devtools.

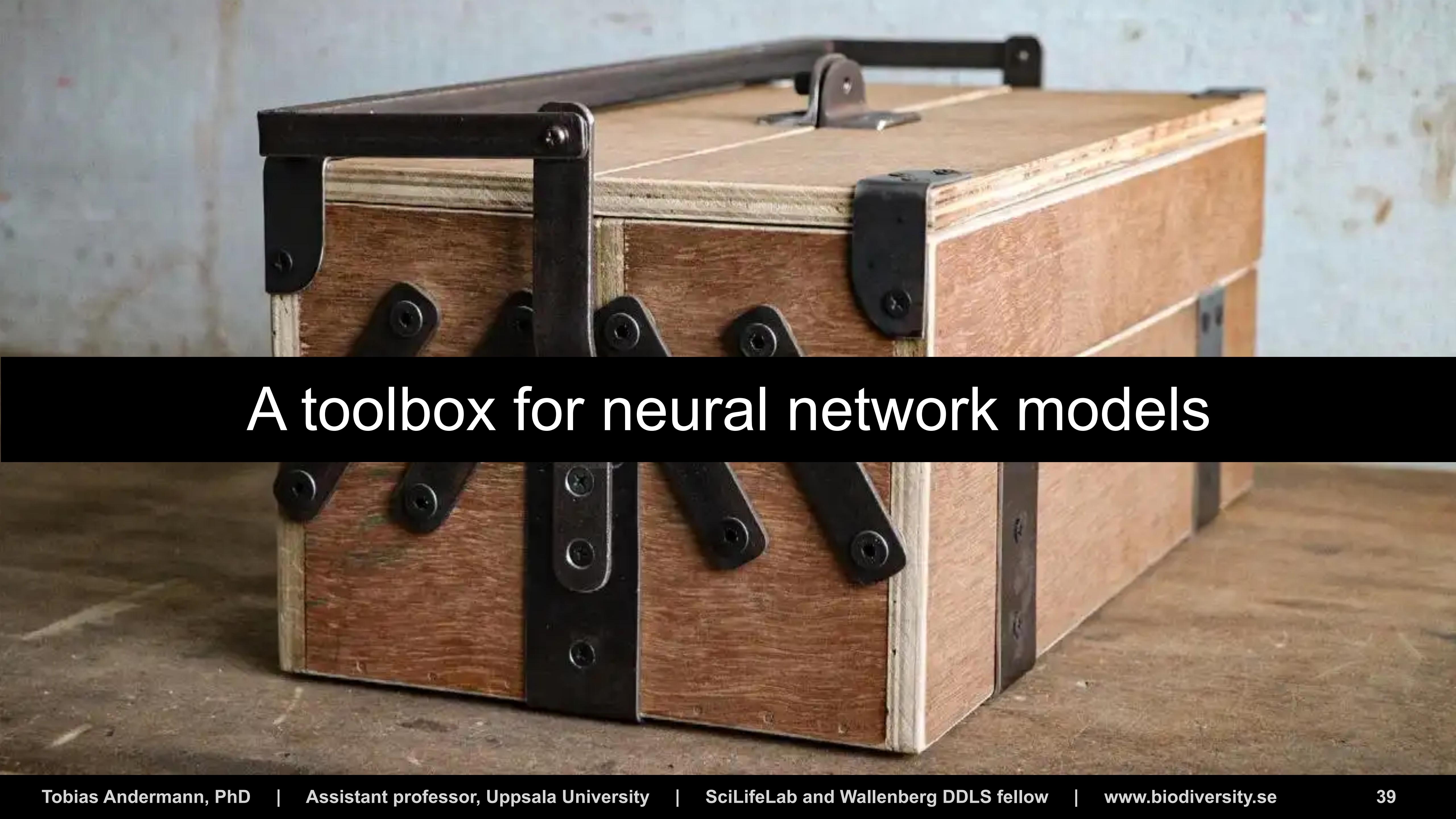
```
install.packages("devtools")
library(devtools)

install_github("IUCNN/IUCNN")
```



Zizka, A., Andermann, T., and Silvestro, D. IUCNN – Deep learning approaches to approximate species' extinction risk. *Diversity and Distributions* 28, 227–241. doi:[10.1111/ddi.13450](https://doi.org/10.1111/ddi.13450).





A toolbox for neural network models

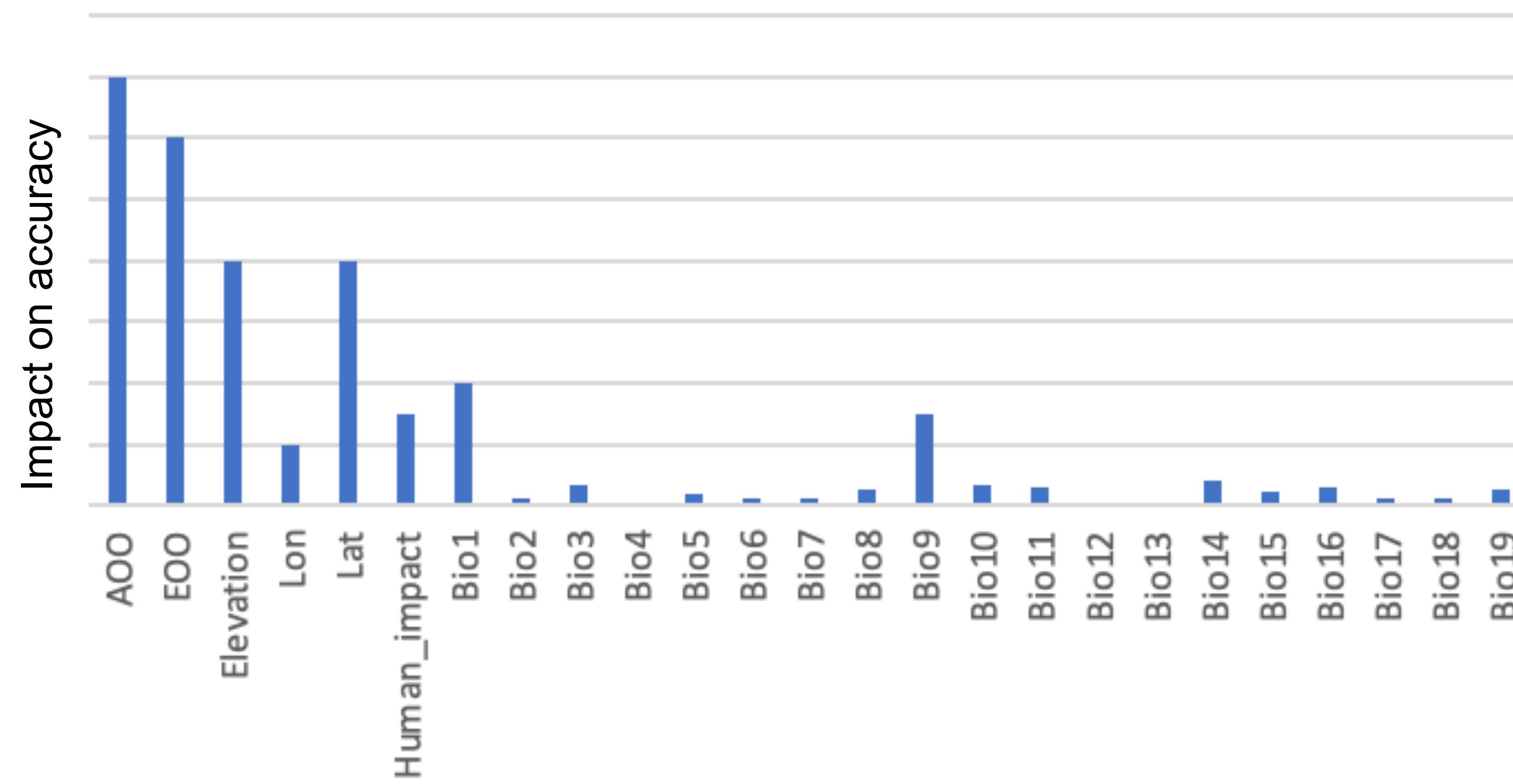
Permutation feature importance

lon	lat	elevation	hfp	bio_1	bio_2	bio_3	bio_4	bio_5	bio_6	bio_7	bio_8	bio_9	bio_10
136.5329	-28.4025	36	3.9215	21.8833	14.5667	45.8071	636.0436	37.9	6.1	31.8	17.4167	28.85	29.55
150.8236	-24.8218	373	5.2002	20.0292	14.5417	54.2599	459.8392	32	5.2	26.8	25.0667	13.7667	25.0667
136.2989	-25.2995	138	0	22.6333	15.6	46.4286	666.822	38.8	5.2	33.6	30.4333	16.0167	30.4333
133.7812	-25.9481	455	6.1402	20.9	14.2167	44.8475	640.3905	36.5	4.8	31.7	27.6833	14.6667	28.4667
141.1054	-17.6605	3	8.8415	27.225	11.6167	55.0553	313.2926	36.2	15.1	21.1	29.4333	24.0333	30.3167
141.7411	-31.0796	171	9.1679	19.3917	14.15	46.3934	616.0388	35.1	4.6	30.5	26.8667	11.6333	26.8667
134.3736	-14.1506	126	0.265	27.3167	13.25	57.8603	301.2424	37.8	14.9	22.9	29.1833	23	30.4833
133.6164	-22.2898	555	0.2611	22.8417	15.8	49.375	592.7203	37.5	5.5	32	29.3167	17.05	29.3167
133.8822	-14.6768	58	0.0152	27.5917	13.4333	56.6807	322.9962	38.3	14.6	23.7	29.4667	22.9833	30.95
141.817	-20.0843	124	3.2501	26.0458	14.7917	54.3811	448.4796	38.1	10.9	27.2	30.5	21.55	30.6
135.409	-15.5381	25	7.776	27.4042	13.425	56.6456	332.9173	38.1	14.4	23.7	30.4333	23.9167	30.9667
140.8475	-17.4635	3	7.8711	26.7333	11.4	54.0284	322.1331	35.6	14.5	21.1	29	23.4	29.9833
137.0116	-17.8959	262	0	26.0458	14.5417	54.6679	411.6401	37.9	11.3	26.6	29.9167	21.8833	30.2167
143.7243	-20.1088	455	3.25	23.8792	14.025	54.1506	432.4374	35.4	9.5	25.9	28.35	19.55	28.35
133.2699	-22.2836	601	0.2681	22.5167	14.5	47.8548	583.1666	36.3	6	30.3	28.85	14.7	28.85
136.5648	-25.7836	101	0	22.7417	15.3333	45.9082	670.7792	39	5.6	33.4	30.6667	16.1167	30.6667
135.4576	-26.4805	126	5.5066	22.2167	14.2	44.795	641.7531	37.8	6.1	31.7	21.9333	15.8833	29.8333
133.6333	-23.8357	606	1.2281	21.2542	14.775	46.4623	618.0742	36.4	4.6	31.8	27.5667	15.2667	28.3
135.83	-16.8239	192	0.2546	26.5917	14.7167	56.3857	382.9659	38.3	12.2	26.1	29.4333	21.2	30.55
150.0949	-23.6582	109	15.603	22.3083	12.7	52.9167	417.3011	32.9	8.9	24	26.85	16.6167	26.85
141.1789	-17.295	10	7.6072	26.9792	11.4417	55.008	299.4576	36	15.2	20.8	29.0167	23.9	30.0167
139.53	-34.32	94	2.3531	16.8875	13.6083	51.9402	471.2031	31.5	5.3	26.2	14	22.4167	22.7167
141.0037	-23.6464	123	0.7295	24.5708	14.9417	48.6699	597.4737	38.6	7.9	30.7	30.5333	16.5667	31.2
135.8615	-28.0871	79	3.2674	22.2167	13.7167	44.9727	623.6671	37.7	7.2	30.5	17.8	22.7	29.7333
135.5817	-26.3667	109	0.4948	22.3875	14.675	45.5745	643.2062	38.1	5.9	32.2	26.35	16	30.0167
134.3555	-14.177	188	0.2518	26.9542	13.175	57.7851	307.3897	37.4	14.6	22.8	28.8833	22.55	30.1667
133.4385	-25.7375	494	2.42	21.05	14.55	45.6113	637.9299	36.7	4.8	31.9	27.75	12.8667	28.5333
134.4207	-14.0479	58	8.7569	27.5667	13.6667	57.9096	298.6129	38.3	14.7	23.6	29.3	23.3	30.7167
142.4779	-20.0137	191	9.7238	25.525	14.5	54.5113	439.615	37.3	10.7	26.6	29.9167	21.1167	29.9833

Test set

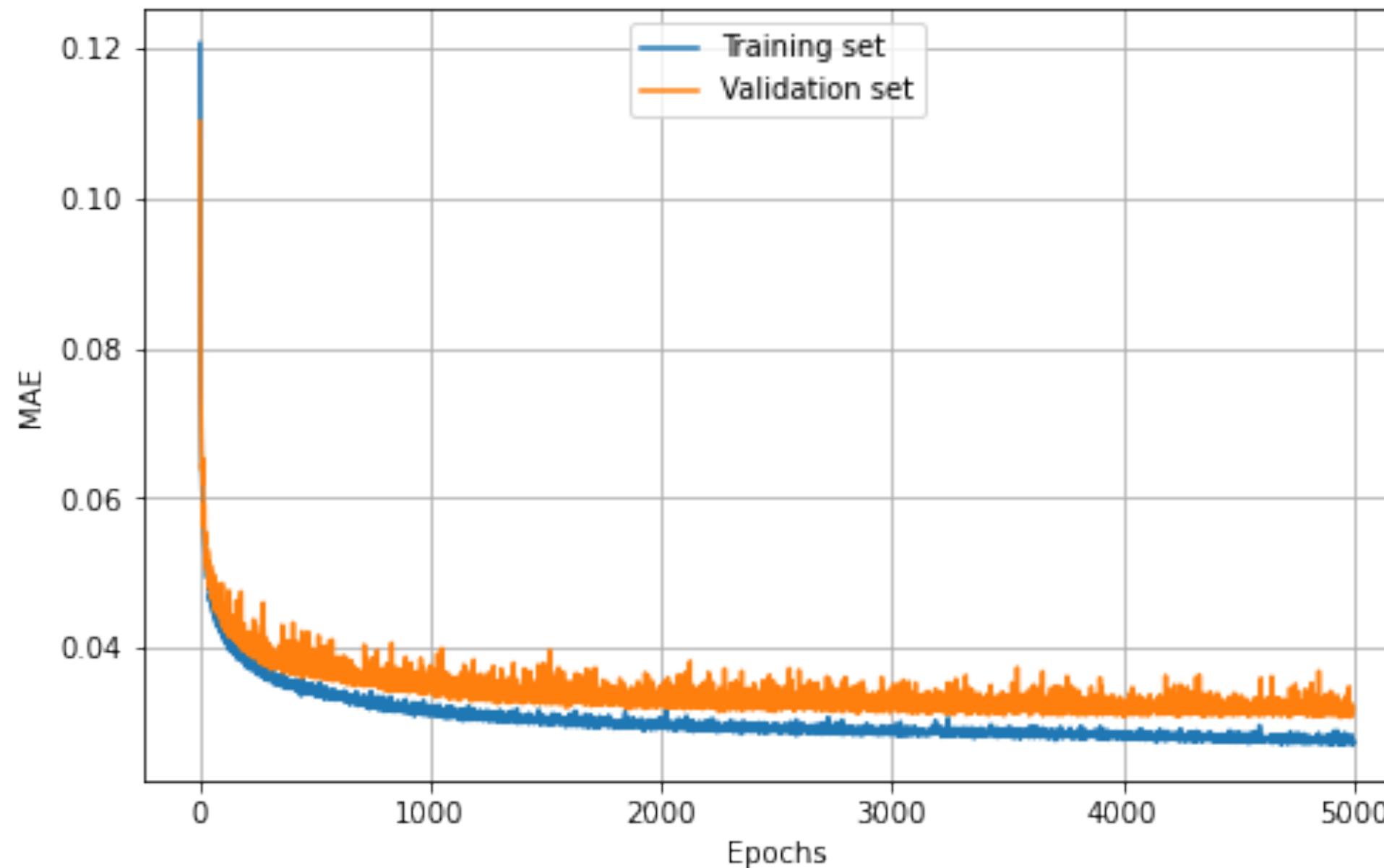
- Predict test set with trained model to establish baseline accuracy
- Shuffle feature column in test set to break information content
- Predict accuracy with modified test set
- Difference in accuracy compared to baseline signifies importance of feature

Feature importance



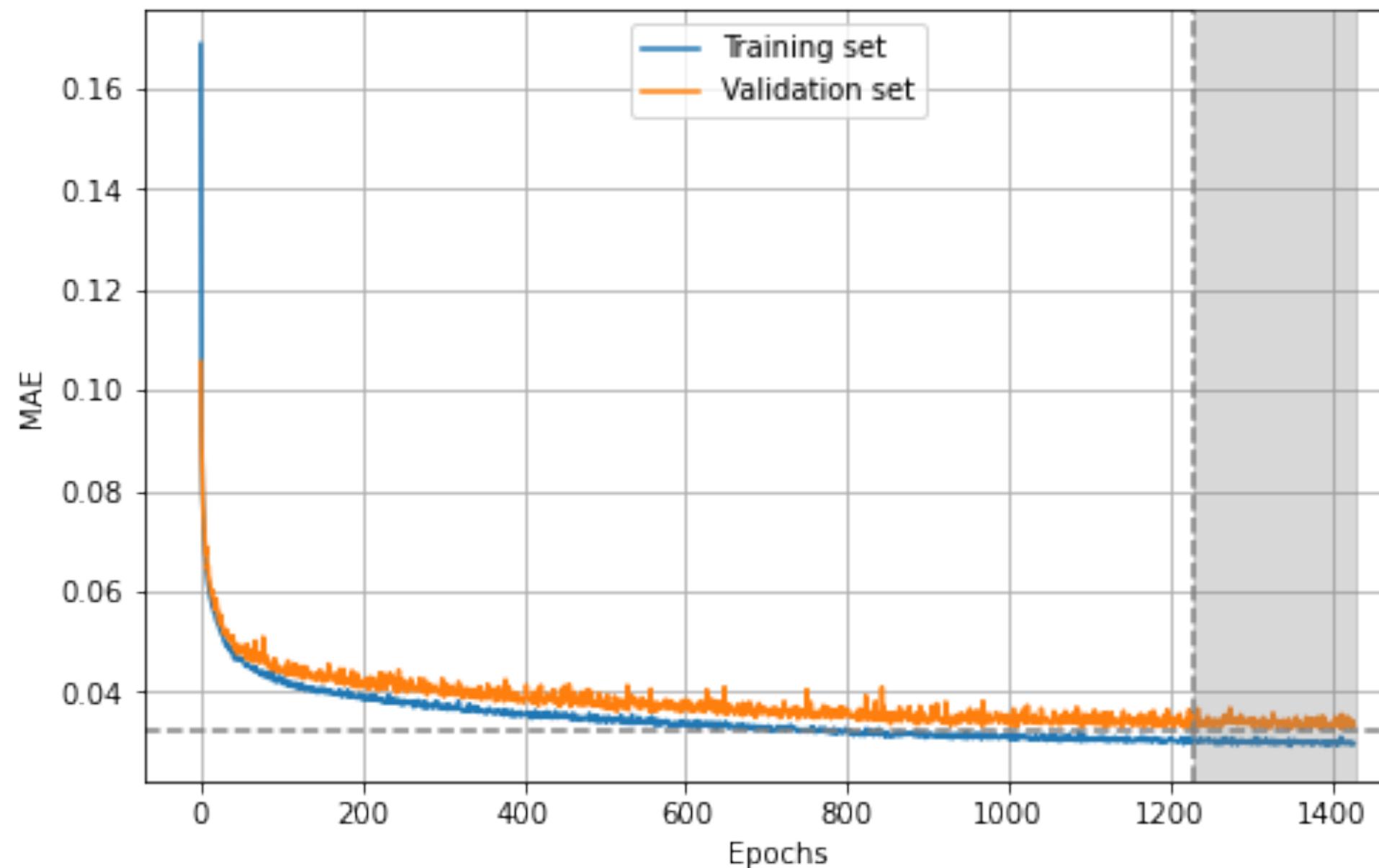
Fit model to data (= training)

```
# Run model training and store training history
history = model.fit(train_features,
                     train_labels,
                     epochs=5000,
                     validation_data=(validation_features, validation_labels),
                     verbose=1,
                     batch_size=40)
```



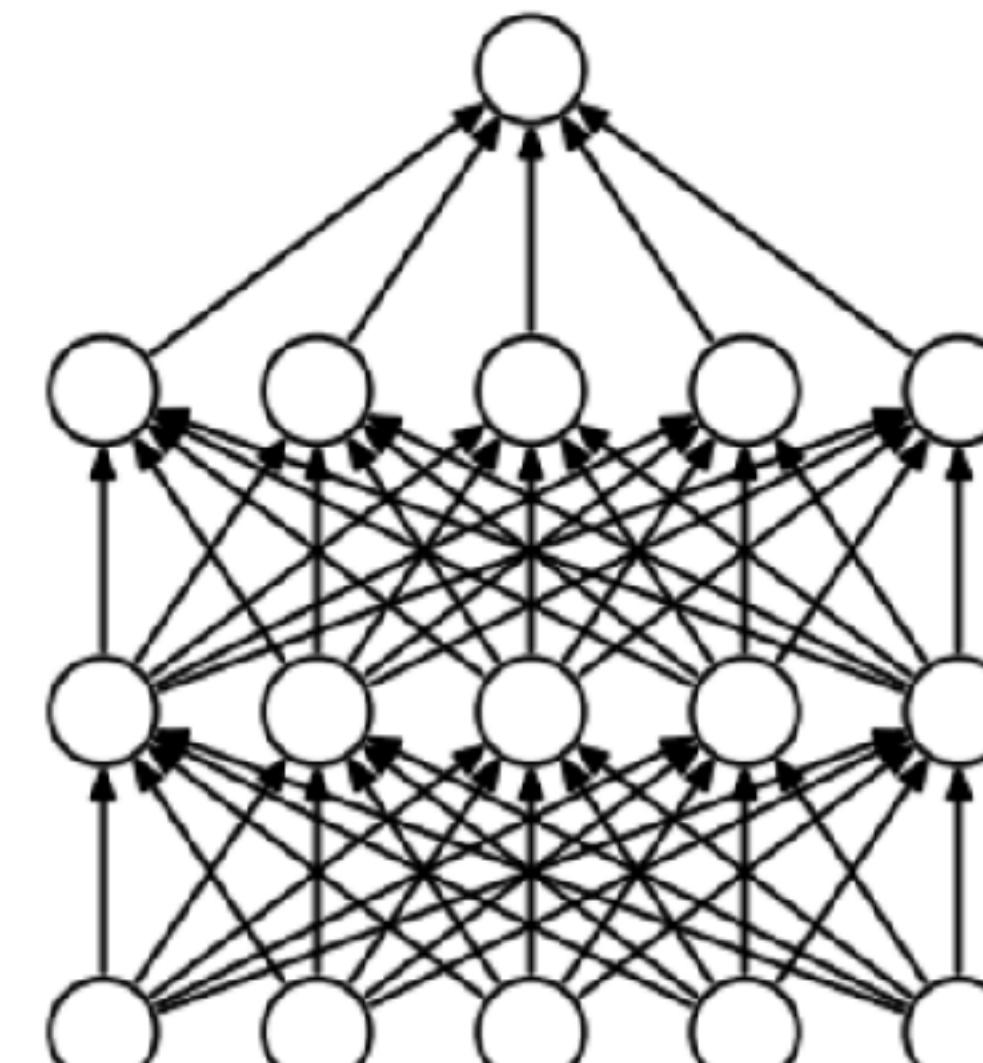
Early stopping

```
# Define early stop of training based on validation set
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_mae', patience=200, restore_best_weights=True)
# Run model training and store training history
history = model.fit(train_features,
                      train_labels,
                      epochs=5000,
                      validation_data=(validation_features, validation_labels),
                      verbose=1,
                      callbacks=[early_stop],
                      batch_size=40)
```

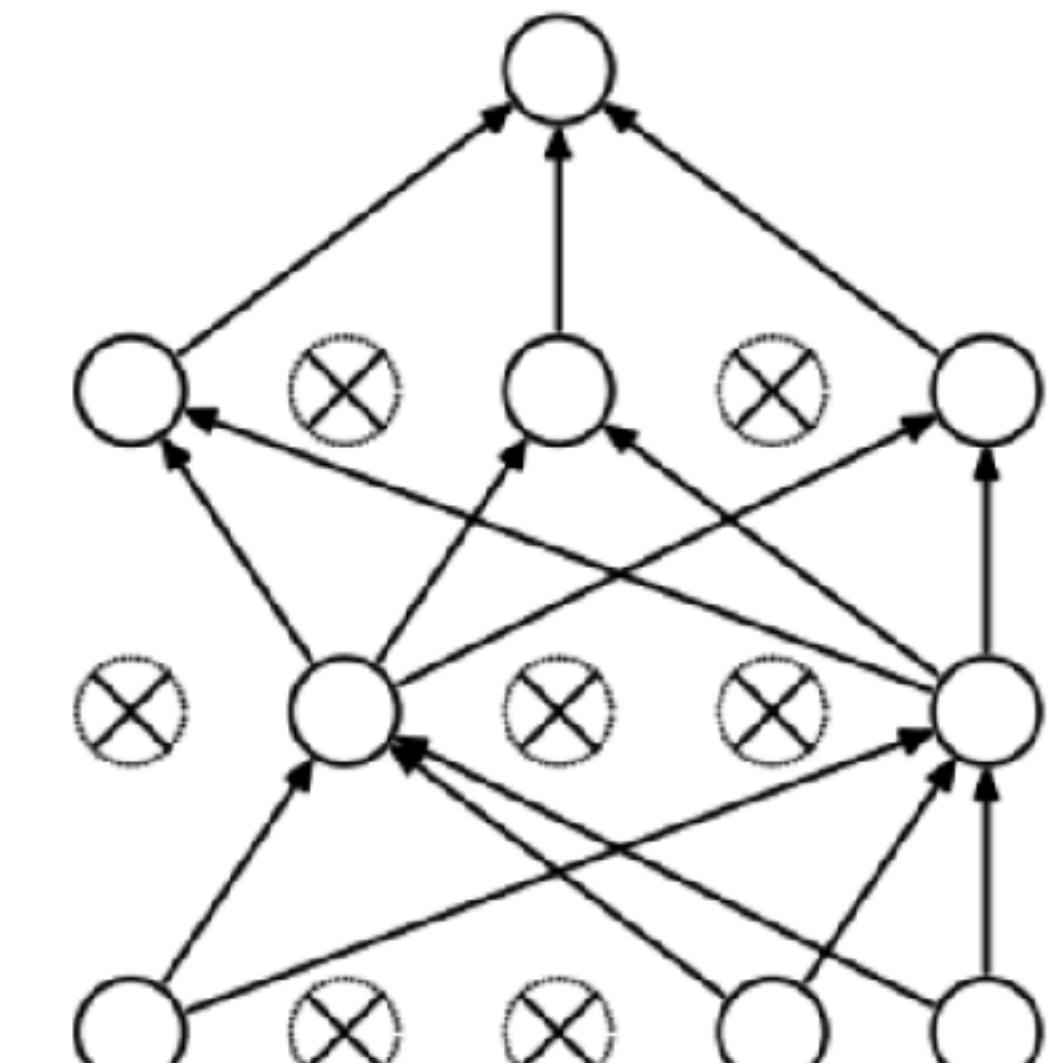


Dropout

- Reduces risk of overfitting
- Every neuron in given layer has a probability of being "dropped out"
- Parameter: dropout rate (usually between 10-50%)



(a) Standard Neural Net

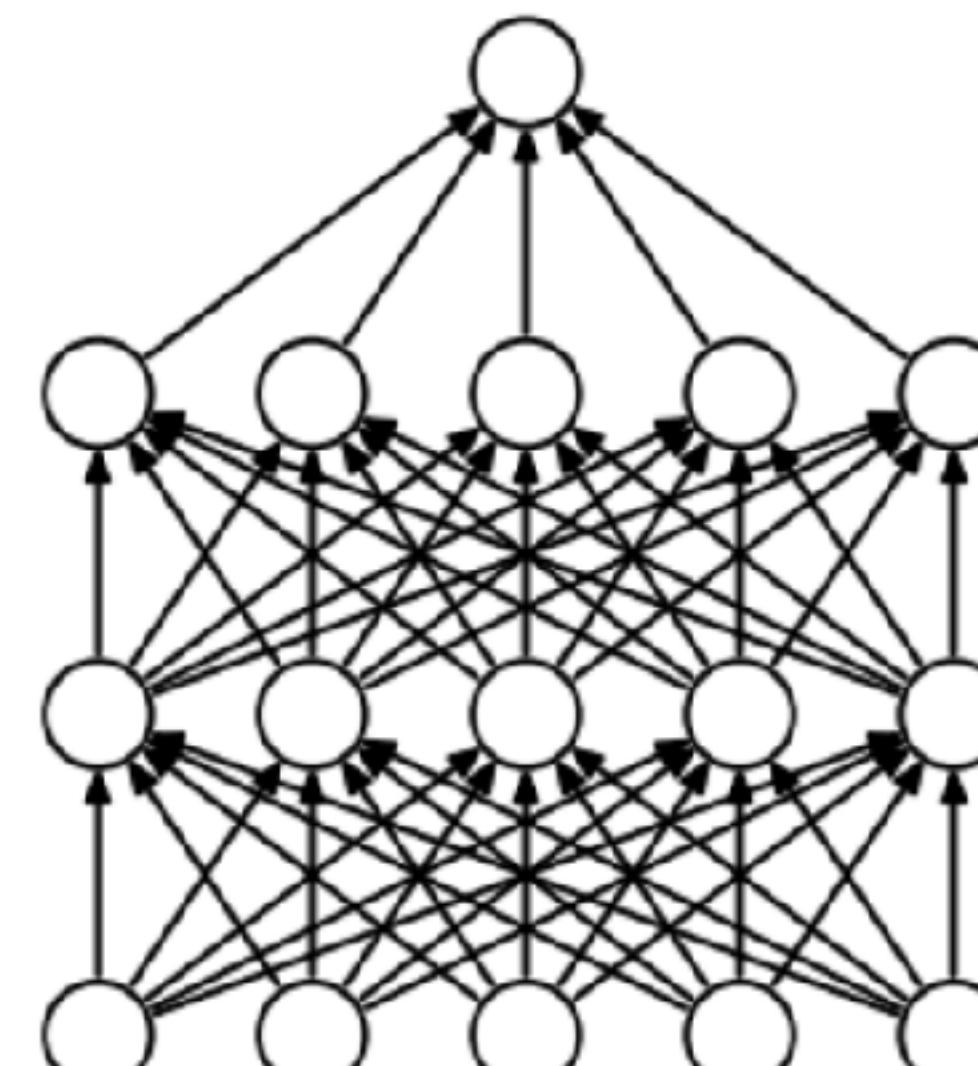


(b) After applying dropout.

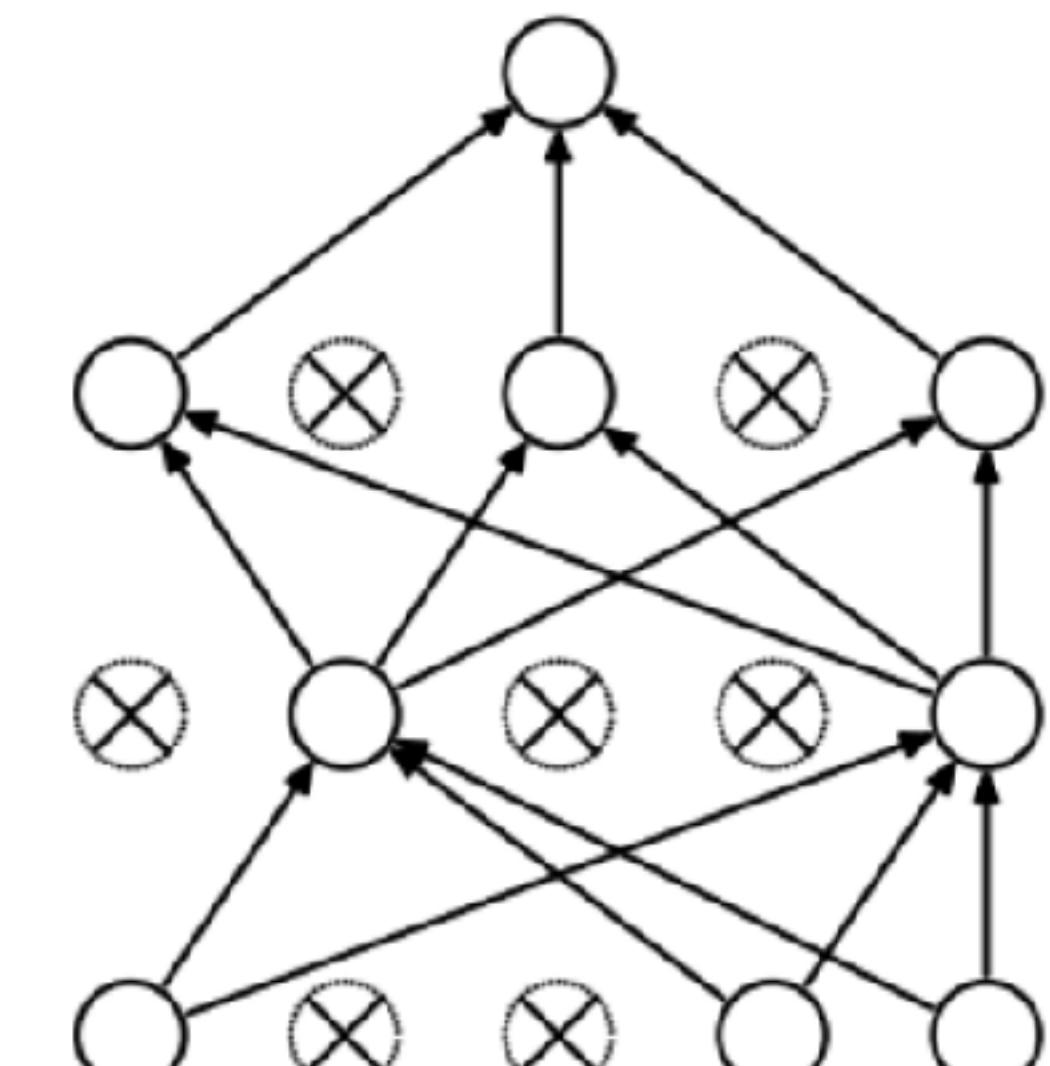
```
architecture = []
# Input layer
architecture.append(tf.keras.layers.Flatten(input_shape=[4]))
# 1st hidden layer
architecture.append(tf.keras.layers.Dense(5, activation='relu', use_bias=False))
architecture.append(tf.keras.layers.Dropout(0.2))
# Output layer
architecture.append(tf.keras.layers.Dense(1, activation='softplus', use_bias=False))
```

Dropout

- Reduces risk of overfitting
- Every neuron in given layer has a probability of being "dropped out"
- Parameter: dropout rate (usually between 10-50%)



(a) Standard Neural Net

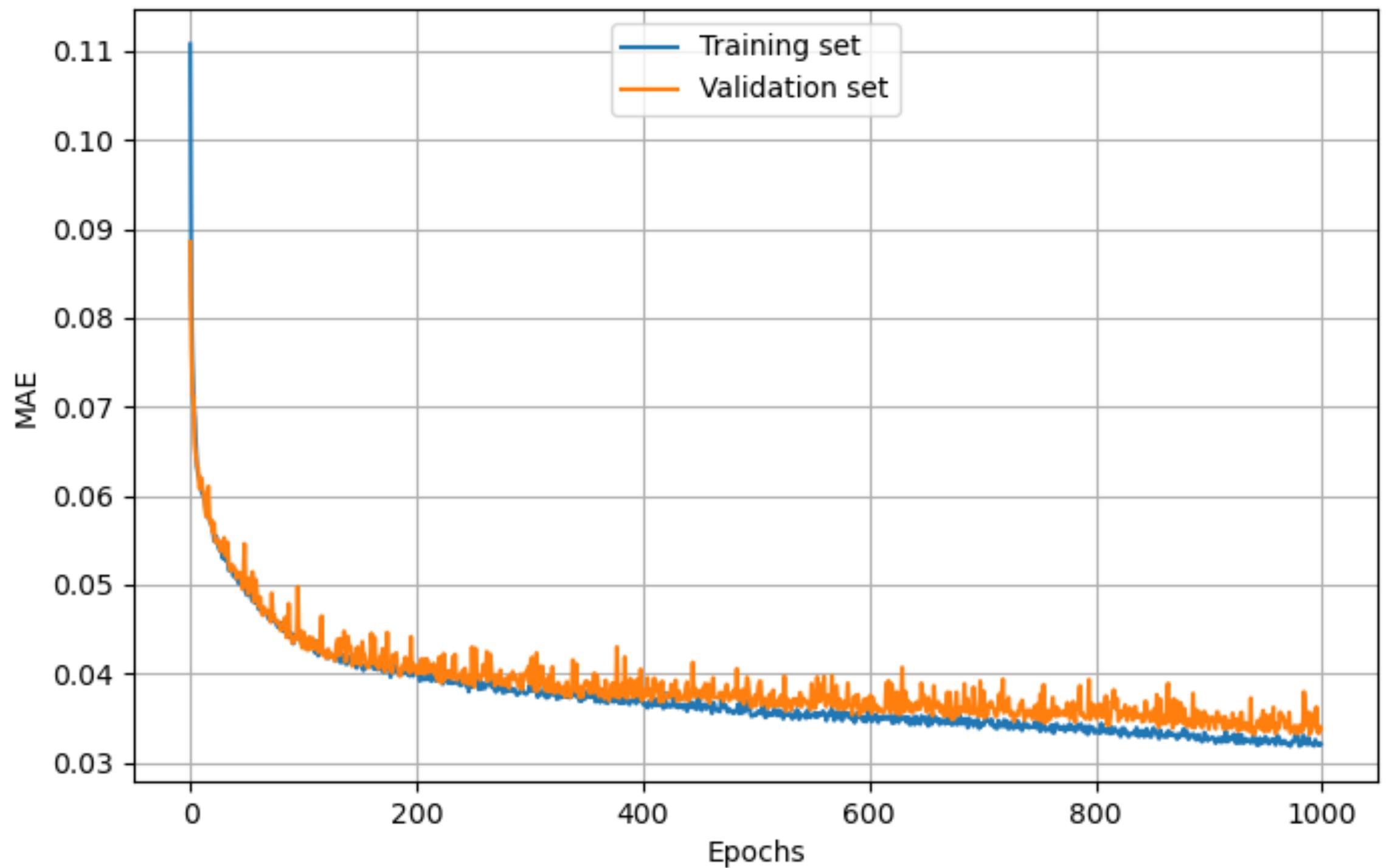


(b) After applying dropout.

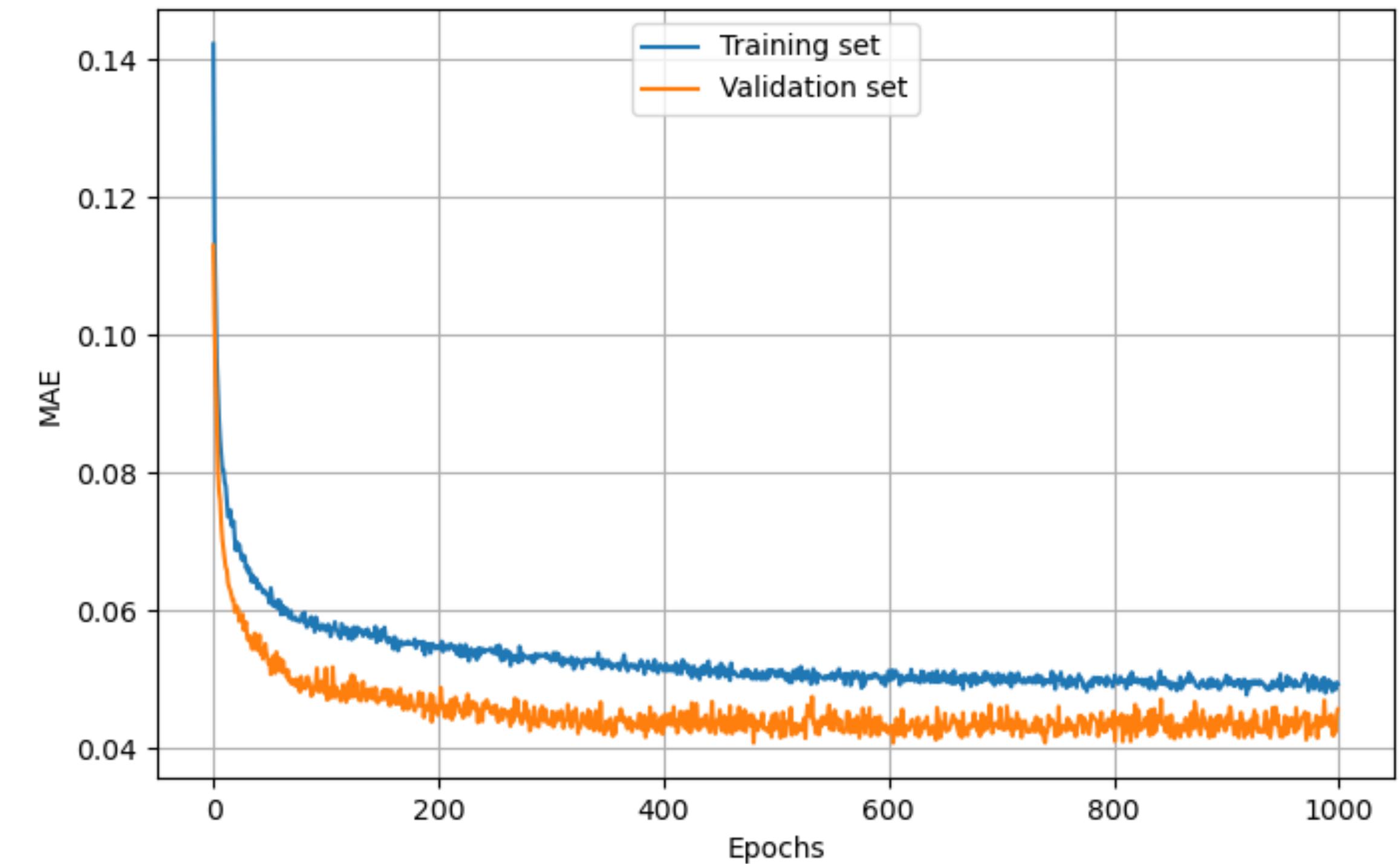
```
architecture = []
# Input layer
architecture.append(tf.keras.layers.Flatten(input_shape=[4]))
# 1st hidden layer
architecture.append(tf.keras.layers.Dense(5, activation='relu', use_bias=False))
architecture.append(tf.keras.layers.Dropout(0.2))
# Output layer
architecture.append(tf.keras.layers.Dense(1, activation='softplus', use_bias=False))
```



Dropout



Regular NN model

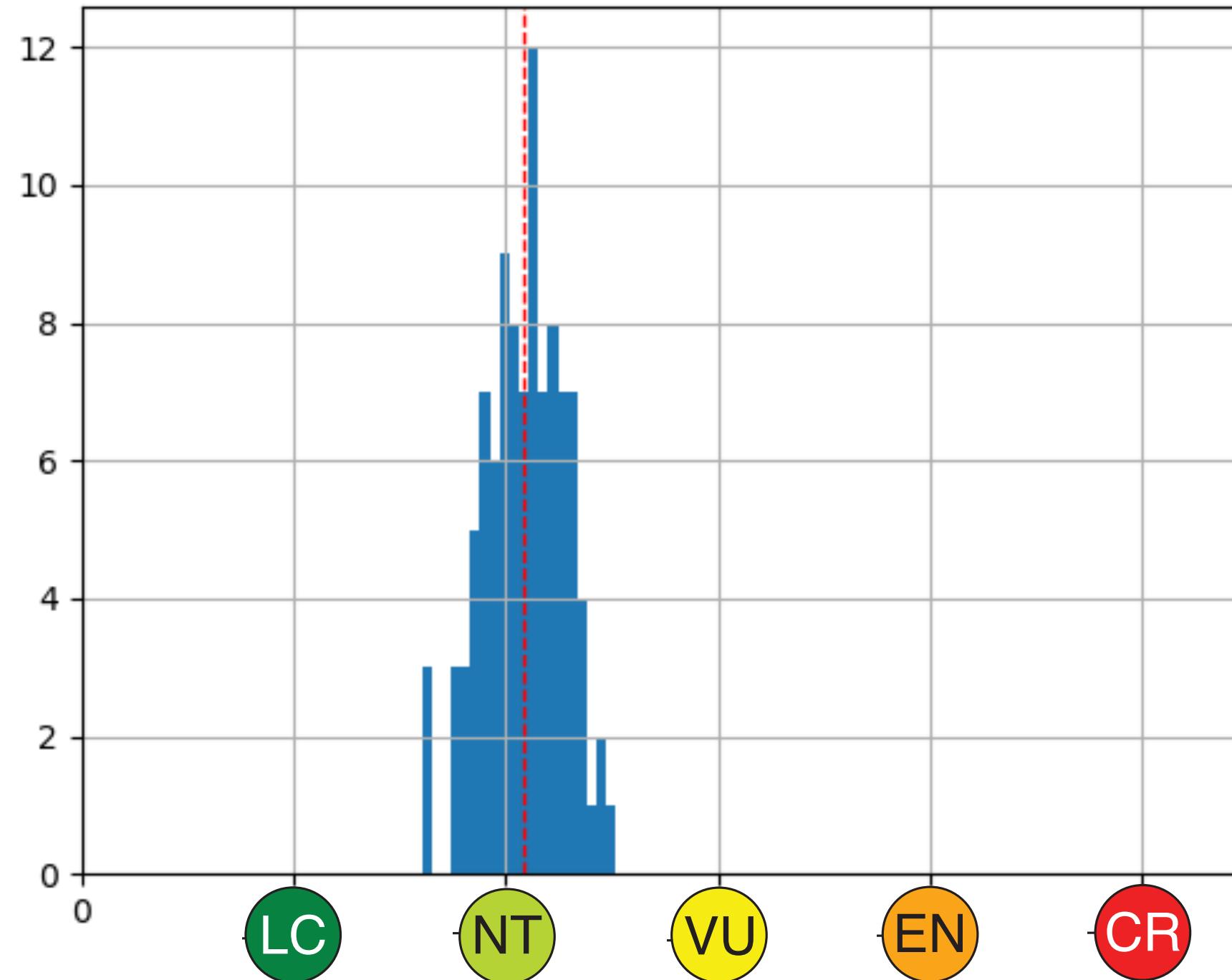


NN model with dropout

Monte Carlo (MC) Dropout

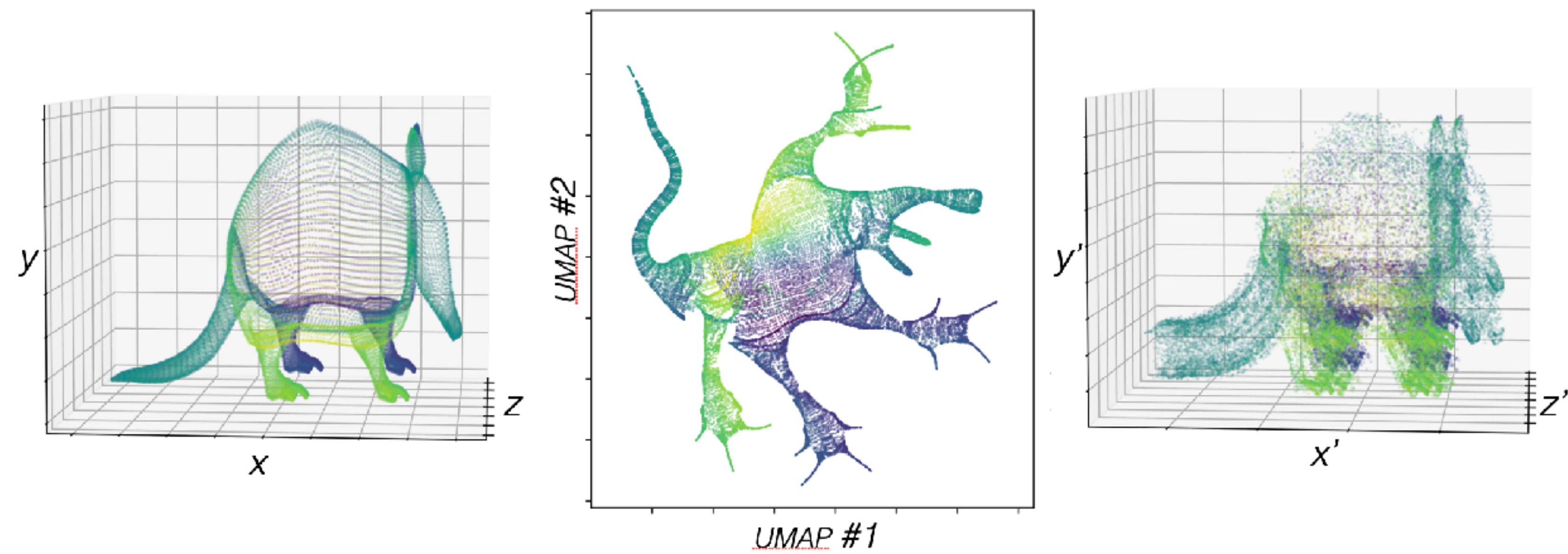
- Method to quantify uncertainty
- Can be applied to any model that was trained with dropout
- Dropout remains active also for predictions, introduces stochasticity
- Repeat predictions e.g. 100 times, quantify uncertainty

100 MC dropout predictions for one test instance



```
mc_dropout_pred = np.stack([model(test_features, training=True) for i in np.arange(100)])
mc_dropout_mean = mc_dropout_pred.mean(axis=0)
mc_dropout_std = mc_dropout_pred.std(axis=0)
```

Dimensionality reduction with UMAP



Purposes:

- Visualize data structure that can't be viewed in original form
- Clustering and outlier detection
- Represent data in fewer dimensions
- Reduce collinearity between axes

Figure provided by Daniele Silvestro

Dimensionality reduction with UMAP

BIO1 = Annual Mean Temperature

BIO2 = Mean Diurnal Range (Mean of monthly (max temp - min temp))

BIO3 = Isothermality (BIO2/BIO7) ($\times 100$)

BIO4 = Temperature Seasonality (standard deviation $\times 100$)

BIO5 = Max Temperature of Warmest Month

BIO6 = Min Temperature of Coldest Month

BIO7 = Temperature Annual Range (BIO5-BIO6)

BIO8 = Mean Temperature of Wettest Quarter

BIO9 = Mean Temperature of Driest Quarter

BIO10 = Mean Temperature of Warmest Quarter

BIO11 = Mean Temperature of Coldest Quarter

BIO12 = Annual Precipitation

BIO13 = Precipitation of Wettest Month

BIO14 = Precipitation of Driest Month

BIO15 = Precipitation Seasonality (Coefficient of Variation)

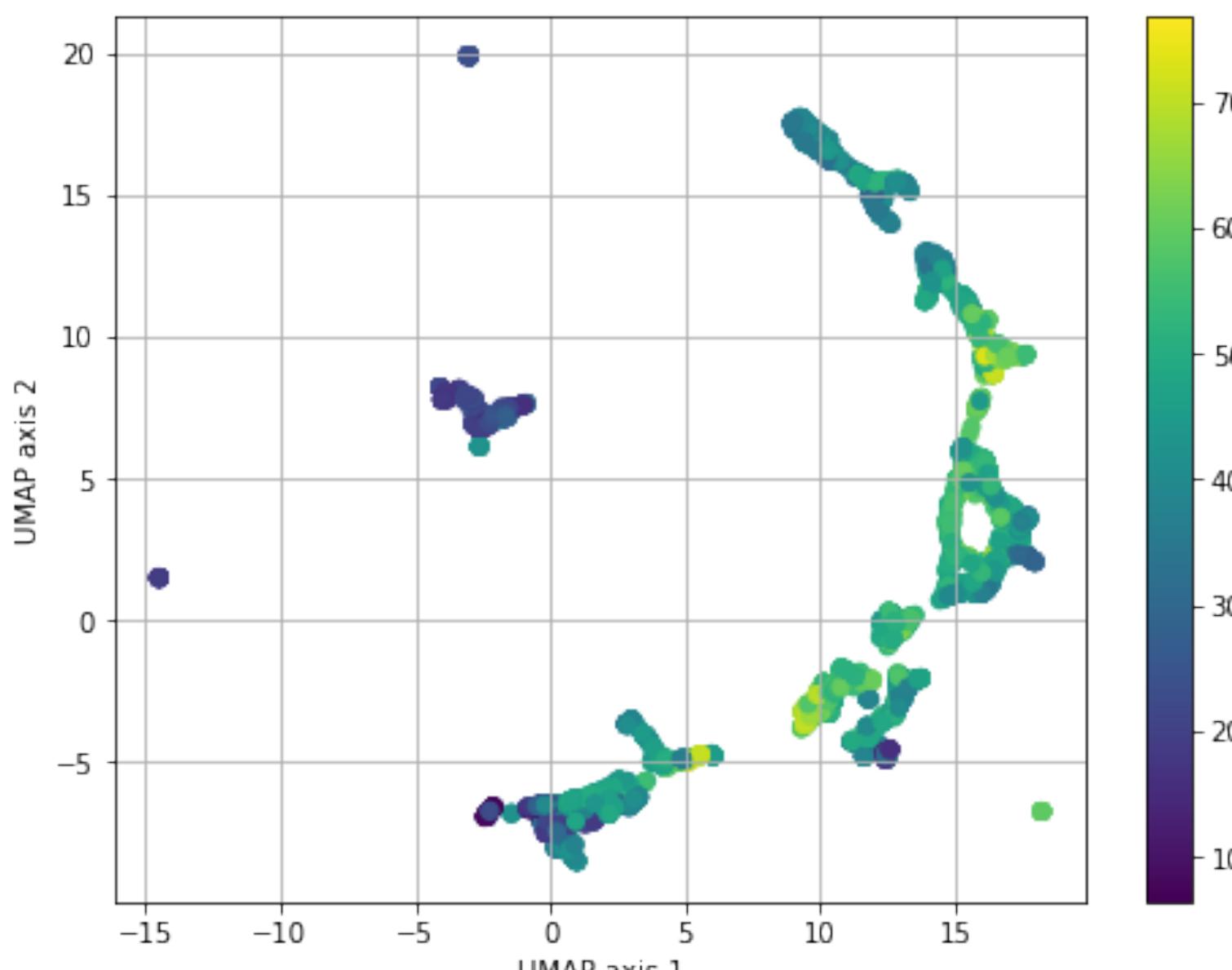
BIO16 = Precipitation of Wettest Quarter

BIO17 = Precipitation of Driest Quarter

BIO18 = Precipitation of Warmest Quarter

BIO19 = Precipitation of Coldest Quarter

```
import umap.umap_ as umap  
  
reducer = umap.UMAP(n_neighbors=250, min_dist=0)  
umap_obj = reducer.fit(bio_features)  
bio_features_transformed = reducer.transform(bio_features)
```



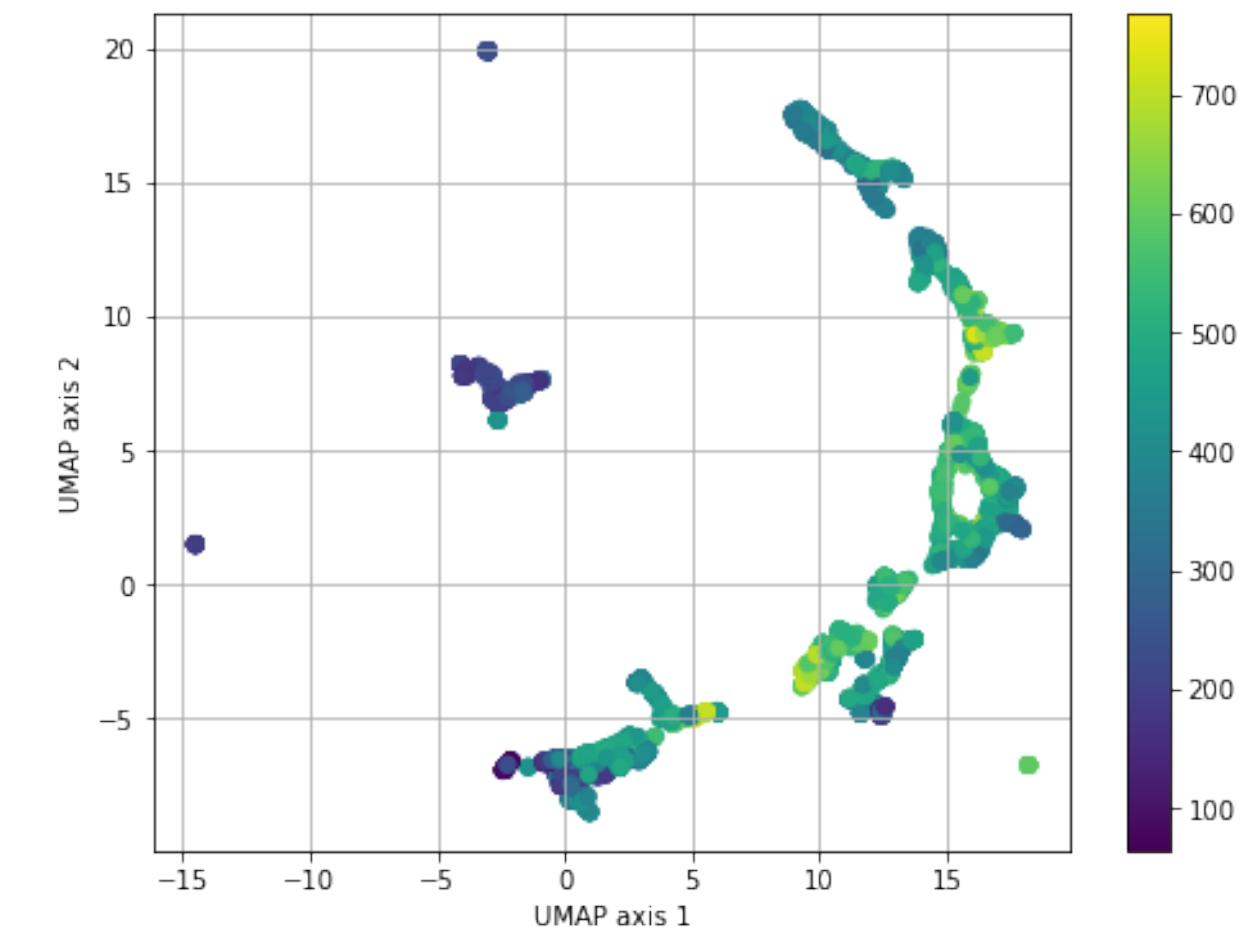
- From 19 to 2 dimensions
- Points colored by species diversity at each site
- UMAP transformed BioClim variables seem to carry some informativeness

BioClim data - highly correlated

Dimensionality reduction with UMAP

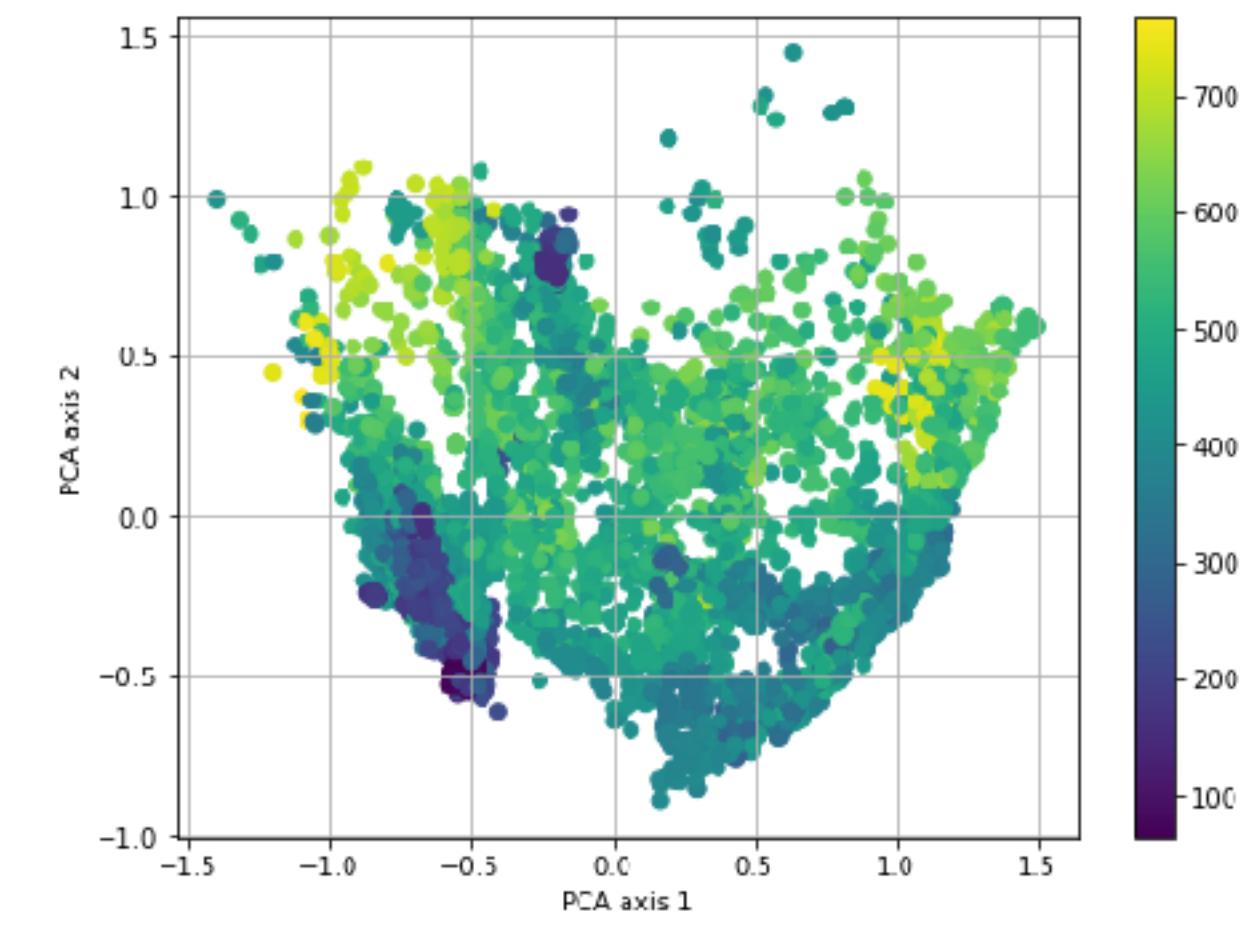
UMAP

```
import umap.umap_ as umap  
  
reducer = umap.UMAP(n_neighbors=250, min_dist=0)  
umap_obj = reducer.fit(biome_features)  
biome_features_transformed = reducer.transform(biome_features)
```



PCA

```
from sklearn.decomposition import PCA  
  
pca = PCA(n_components=2)  
pca.fit(biome_features)  
biome_features_pca_transformed = pca.transform(biome_features)
```

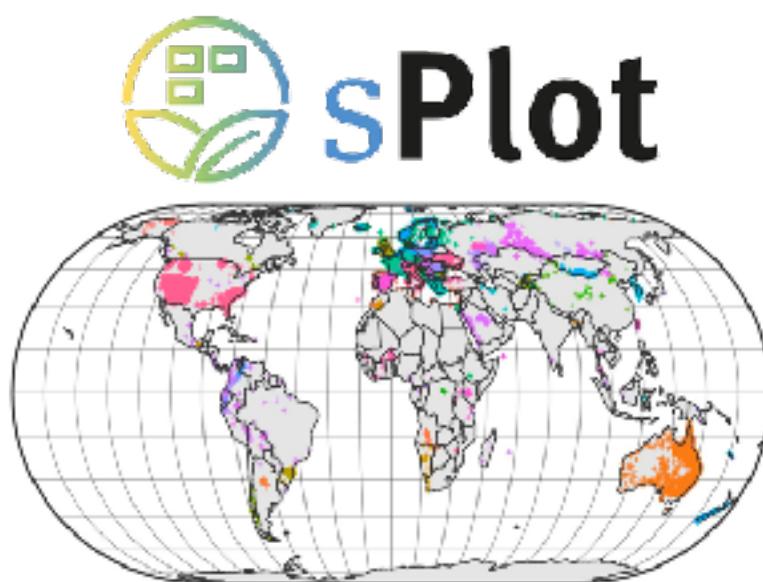




Modeling species richness

Modeling species richness

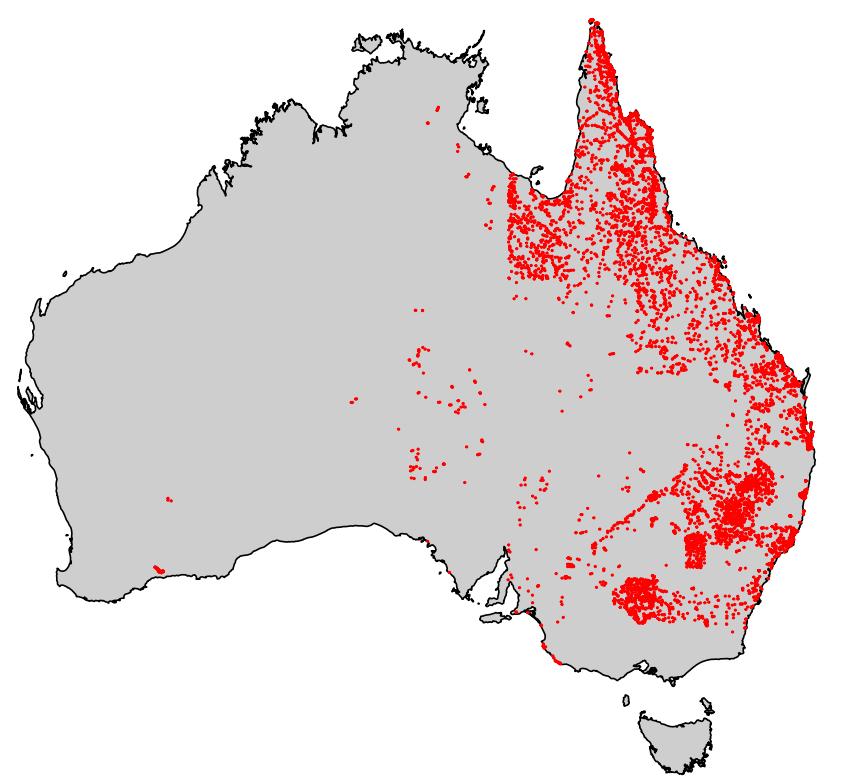
Extrapolating from point estimates



- Train neural network (**machine learning**) with vegetation plot data (species inventory)
- Point measurements of plant species richness



Estimating Alpha, Beta, and Gamma Diversity Through Deep Learning

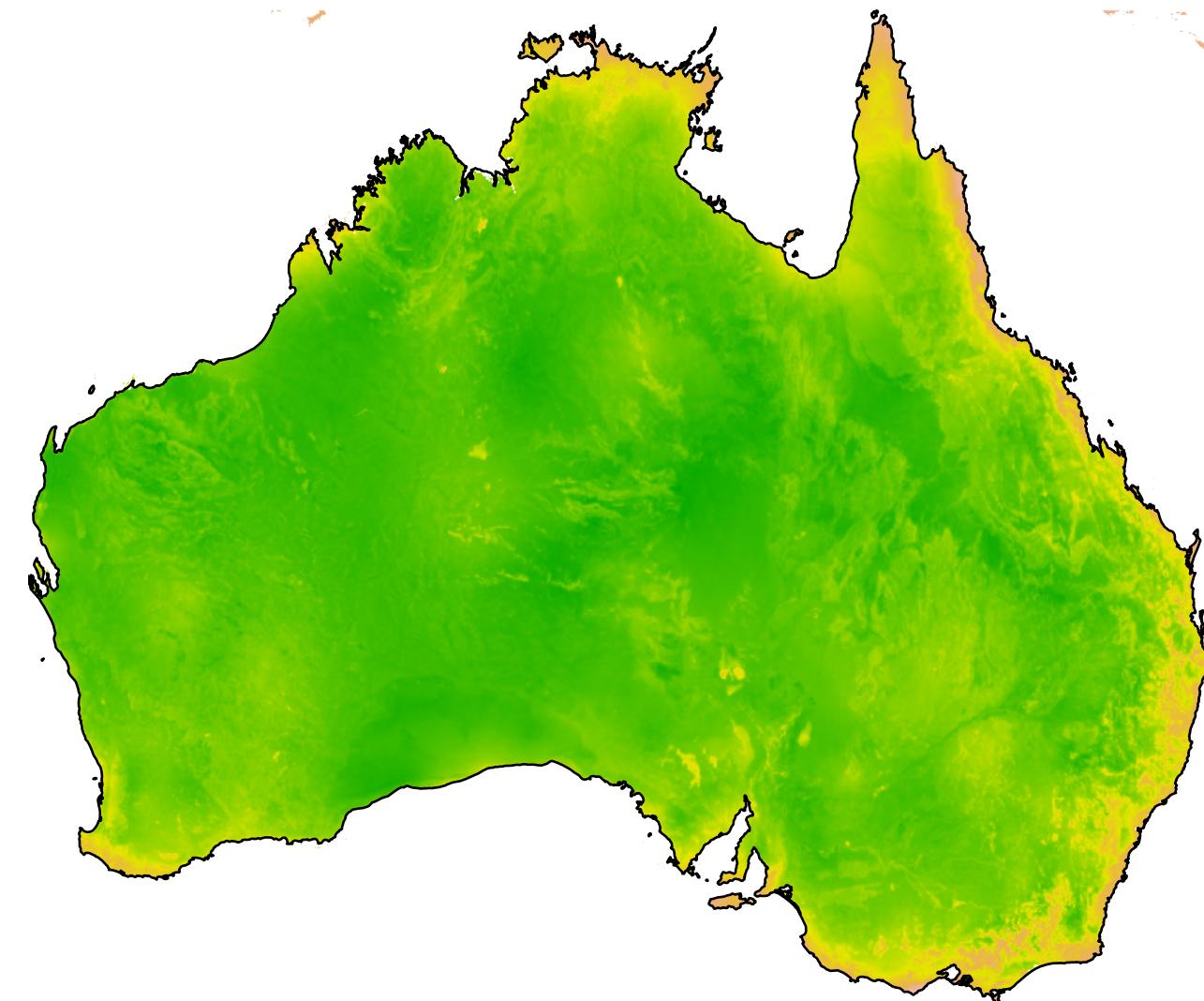


Tobias Andermann^{1,2,3,4*}, Alexandre Antonelli^{1,2,5,6}, Russell L. Barrett^{7,8} and

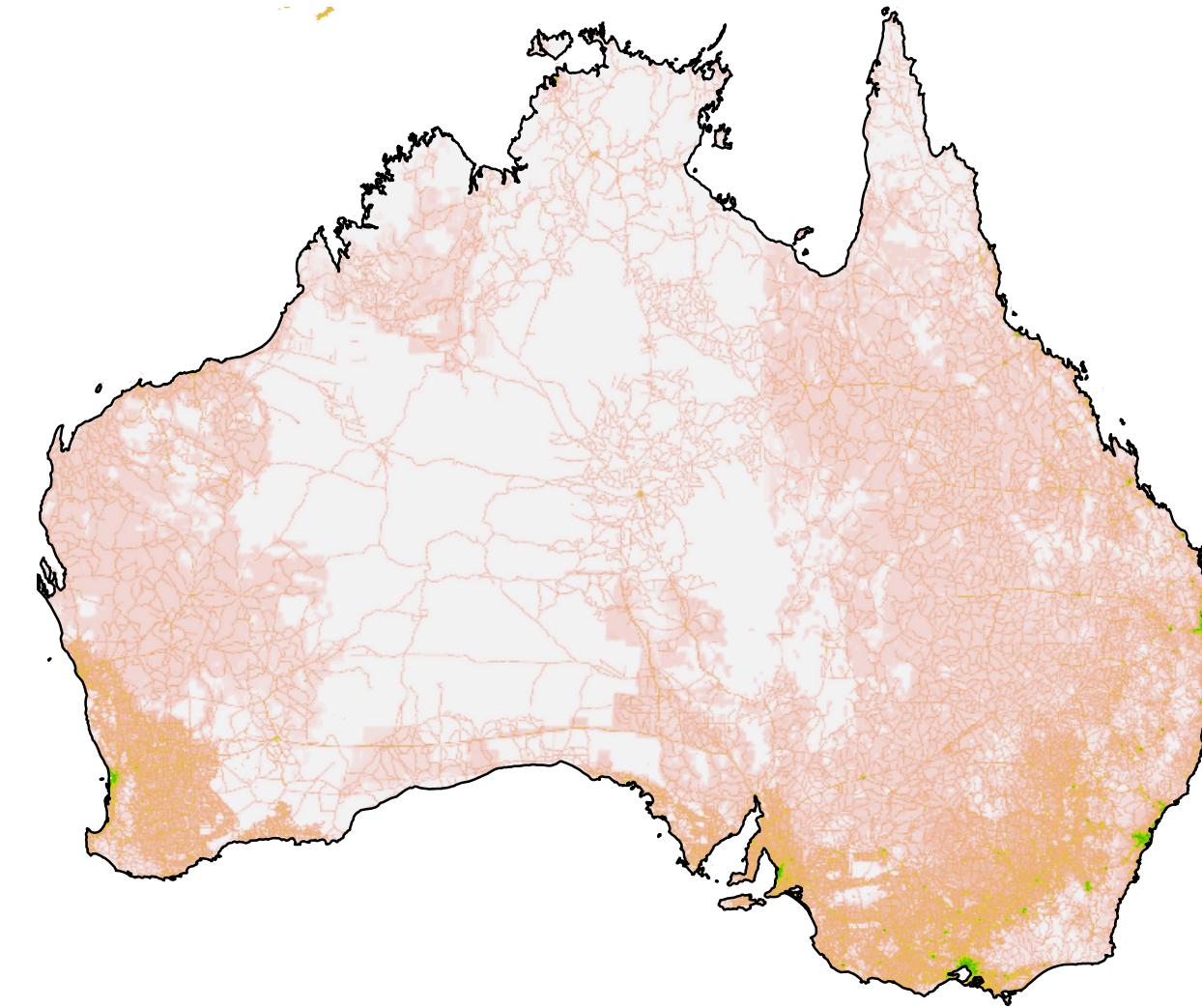


Daniele Silvestro^{1,2,3,4}

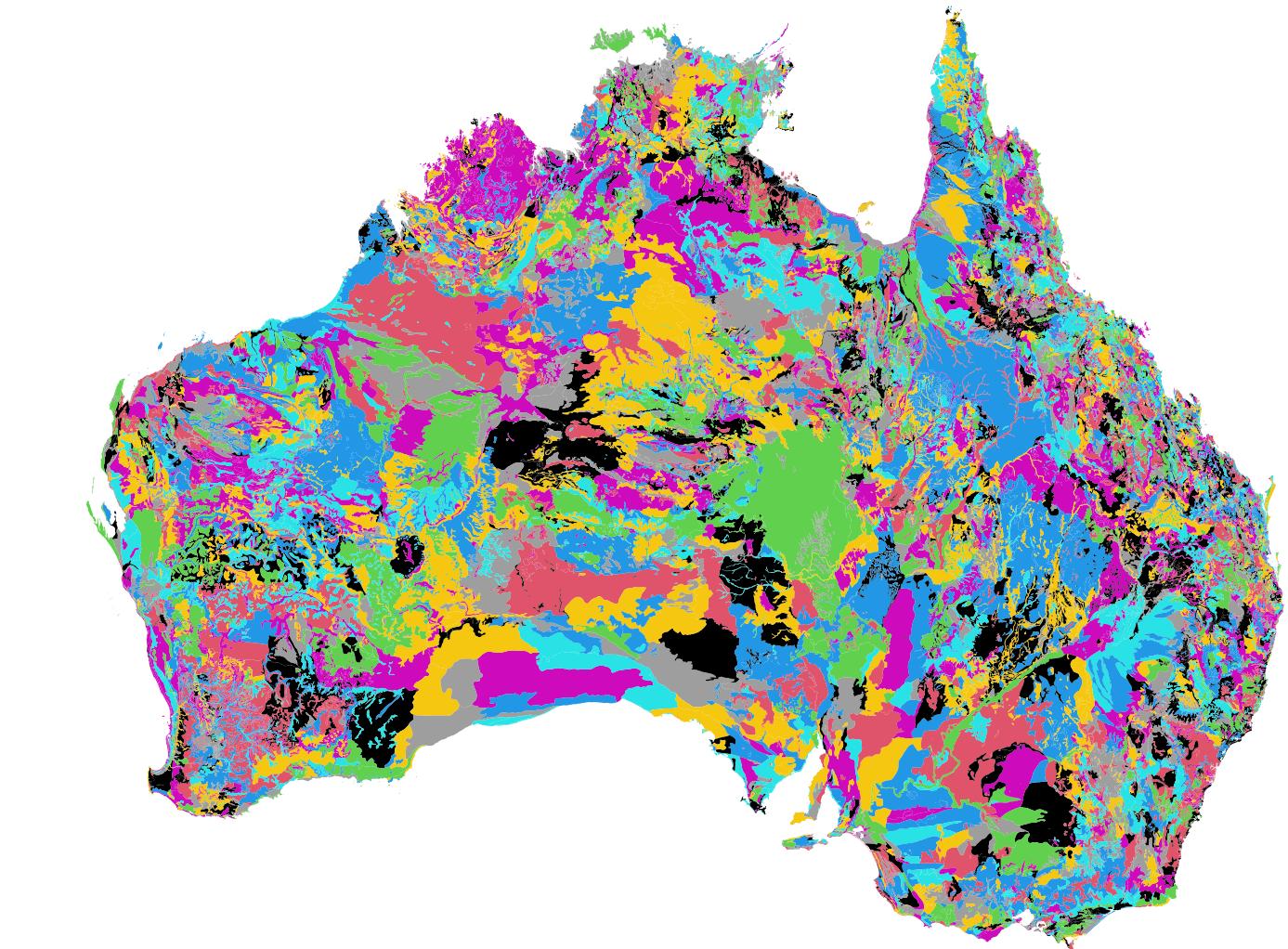
Spatially explicit predictors



climatic variables, e.g.
Bioclim (BIO1-BIO19)



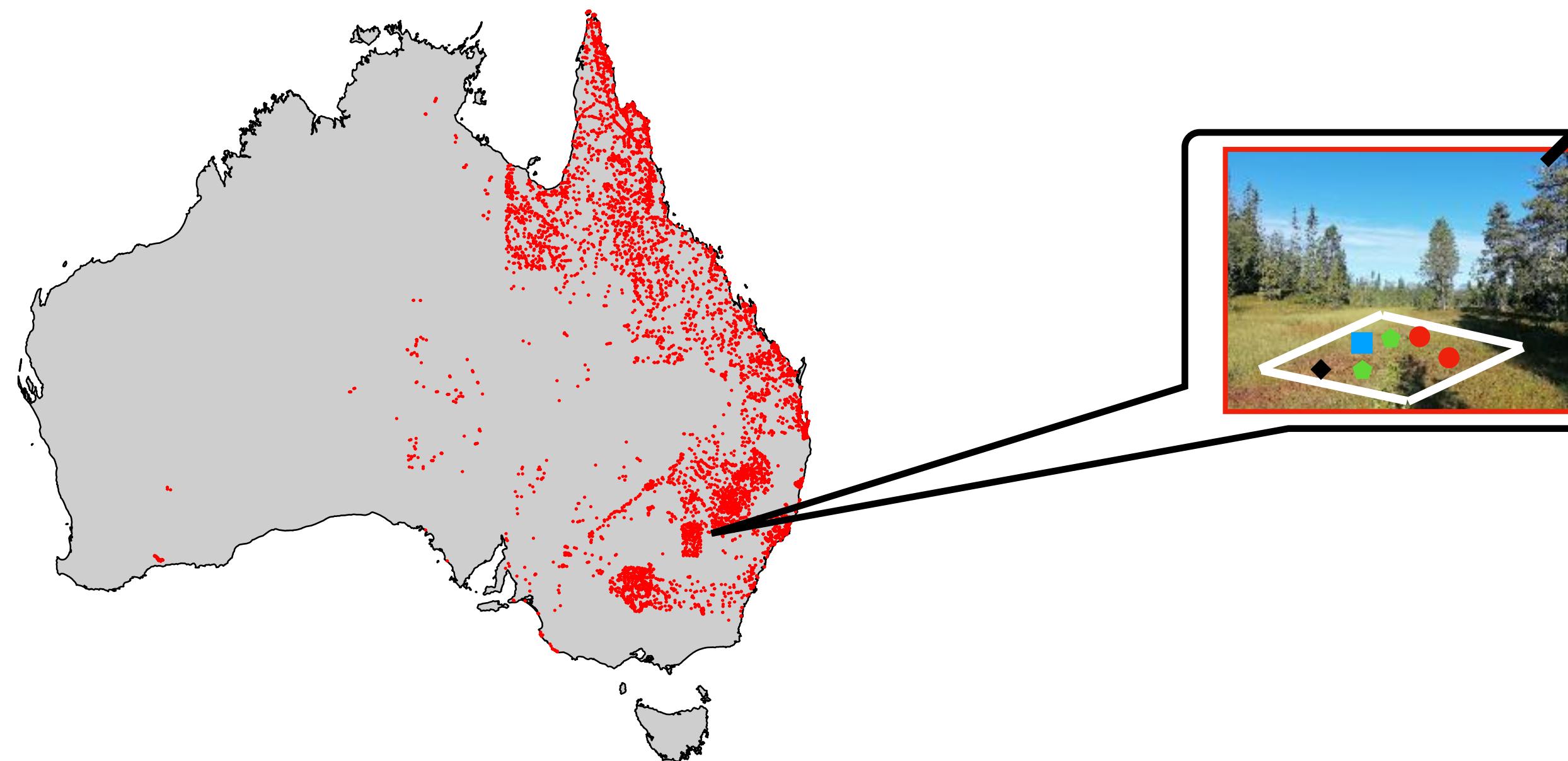
human footprint



soil type

Andermann et al. (2022). *Frontiers in Plant Science*

Preparing training data

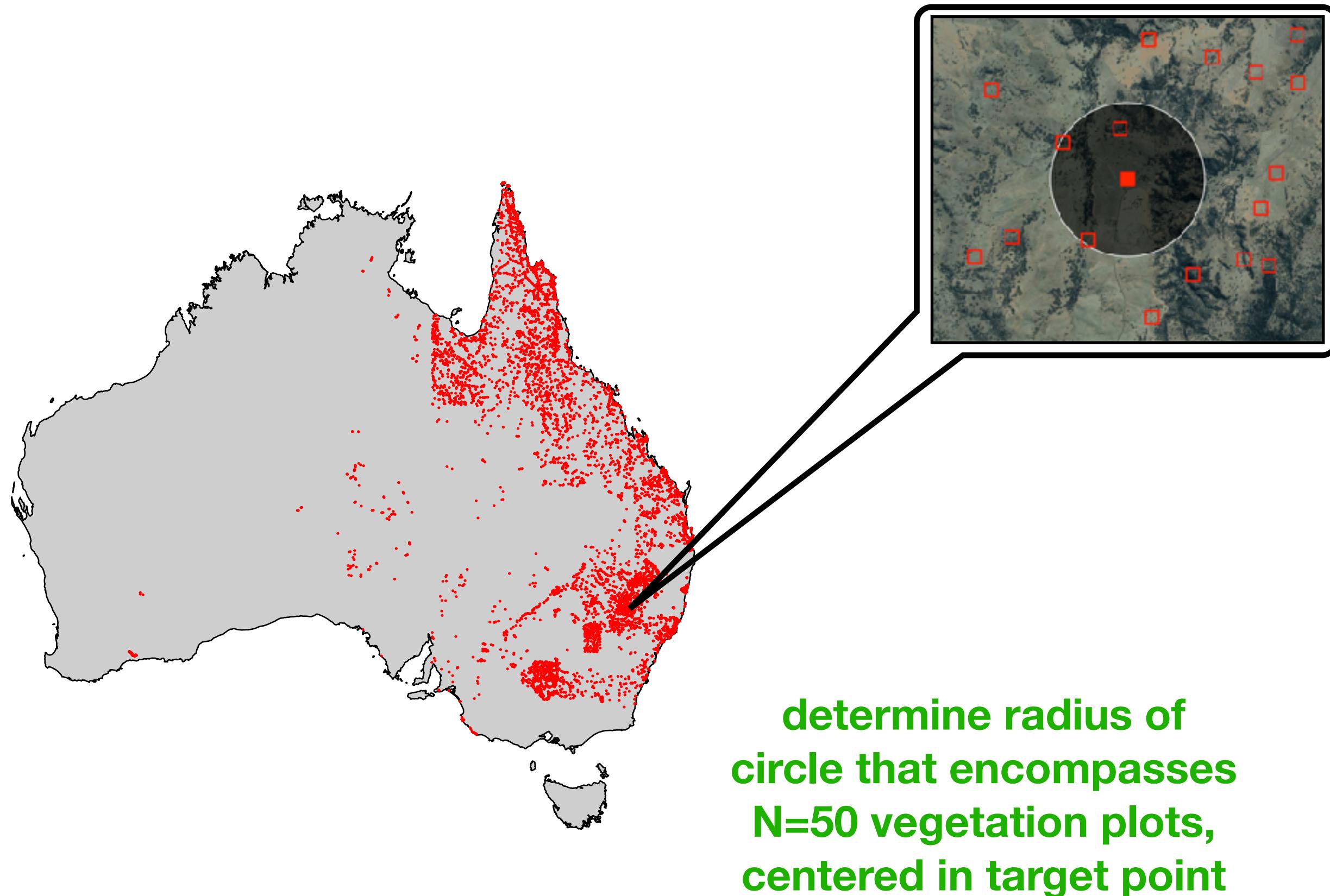


Site A:

local species richness: 32
size of area: 20m²
longitude: 146.23
latitude: -34.52
avg. temperature: 20
avg. rainfall: 500
human footprint: 0.94
soil category: 12
...

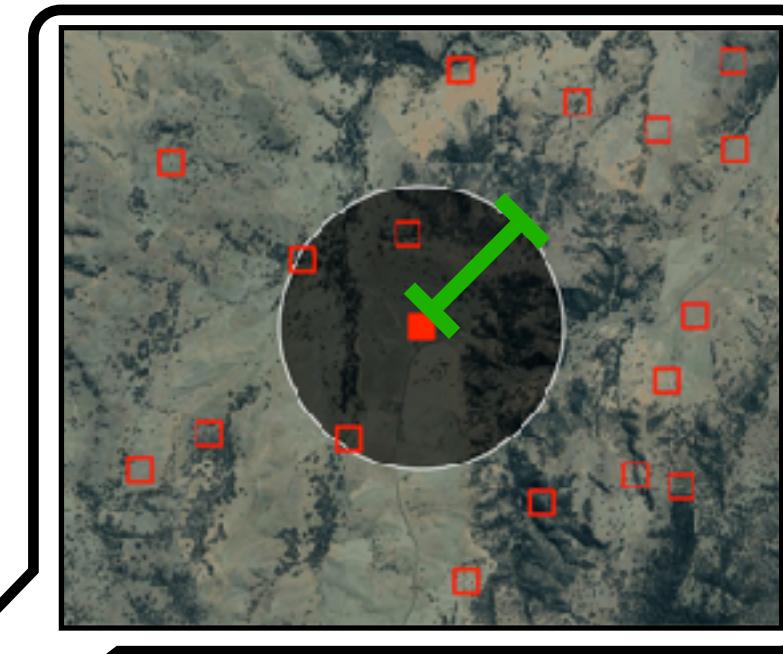
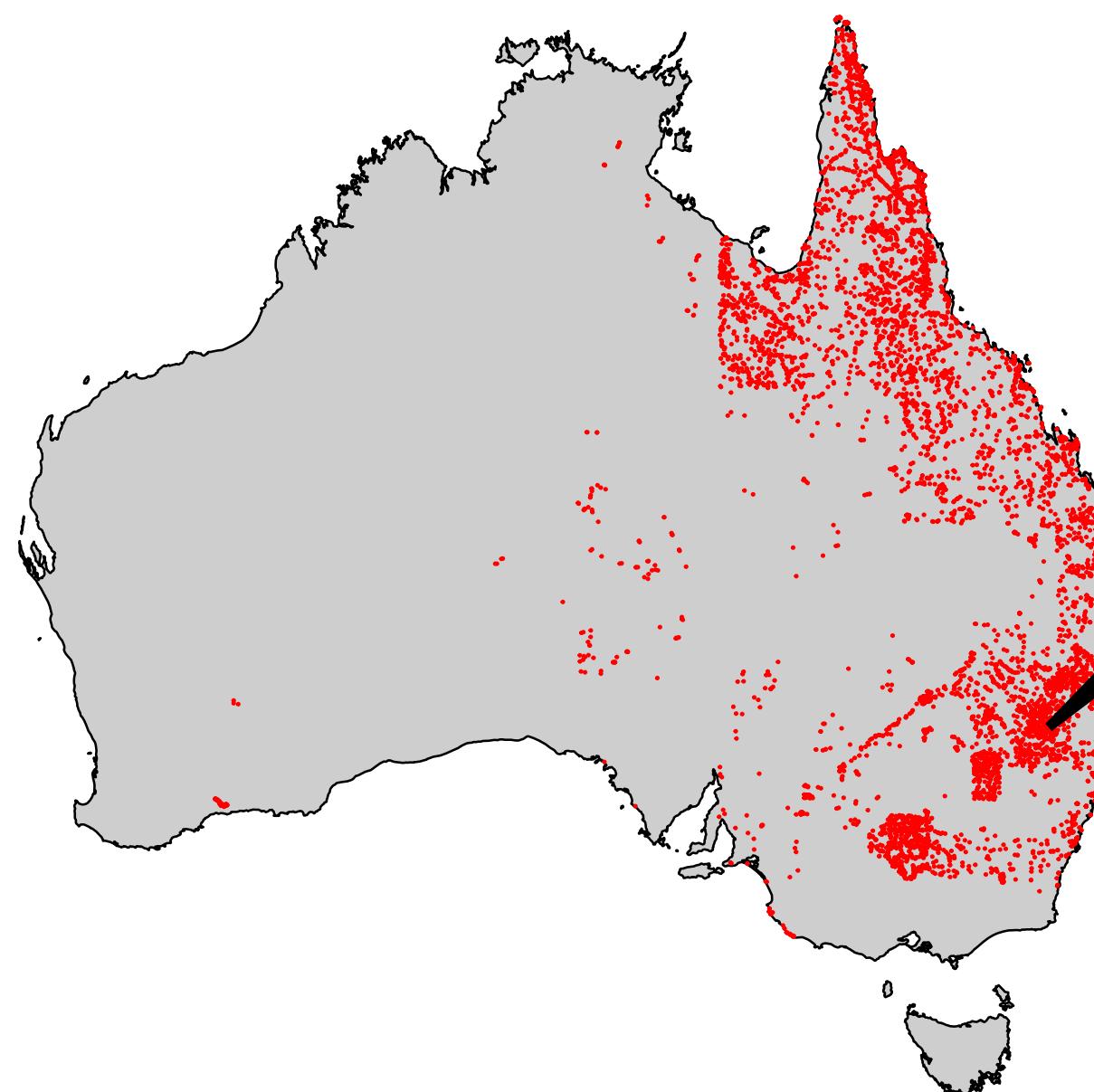
Andermann et al. (2022). *Frontiers in Plant Science*

Preparing training data



Andermann et al. (2022). *Frontiers in Plant Science*

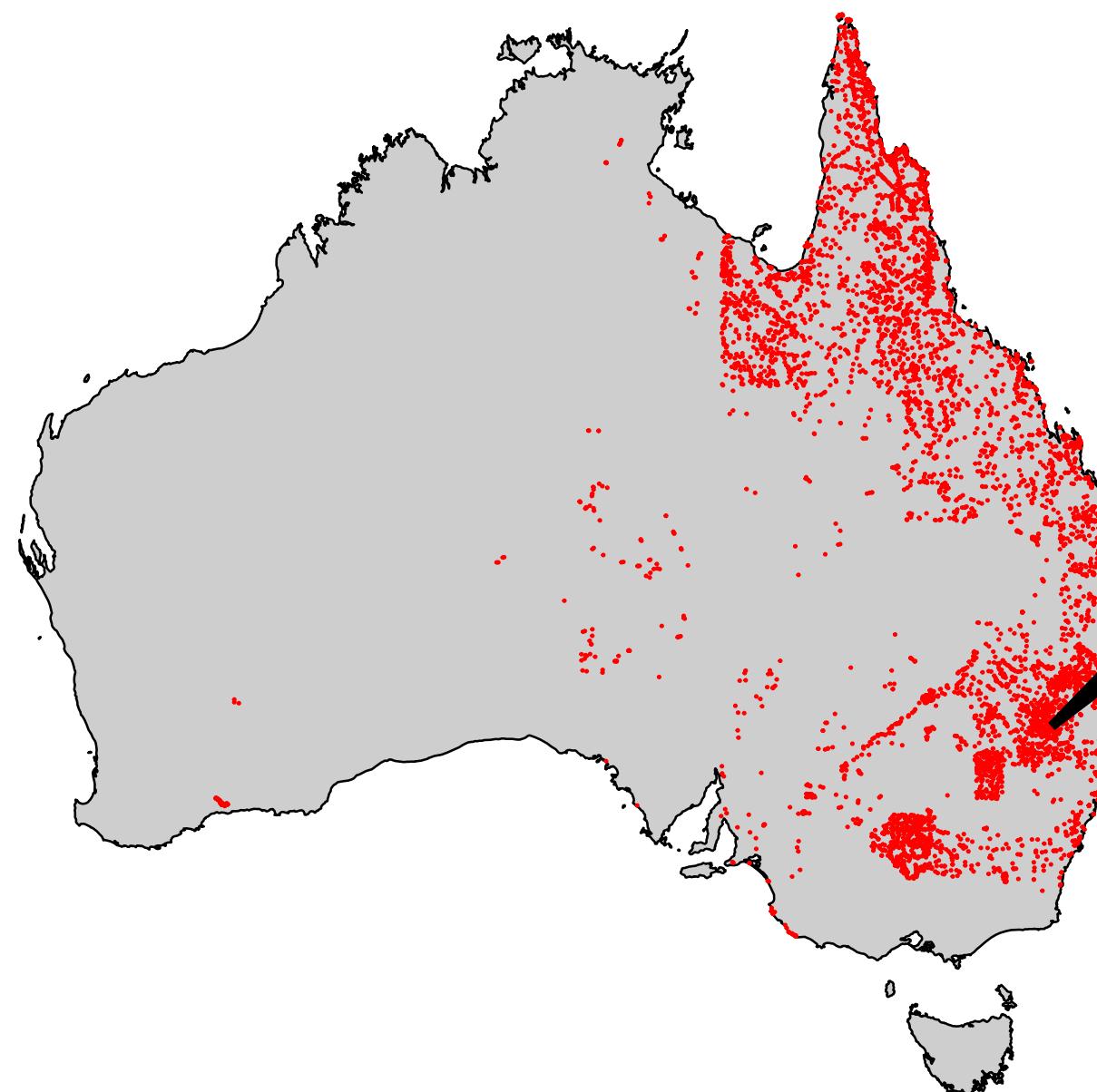
Preparing training data



determine radius of
circle that encompasses
N=50 vegetation plots,
centered in target point

Andermann et al. (2022). *Frontiers in Plant Science*

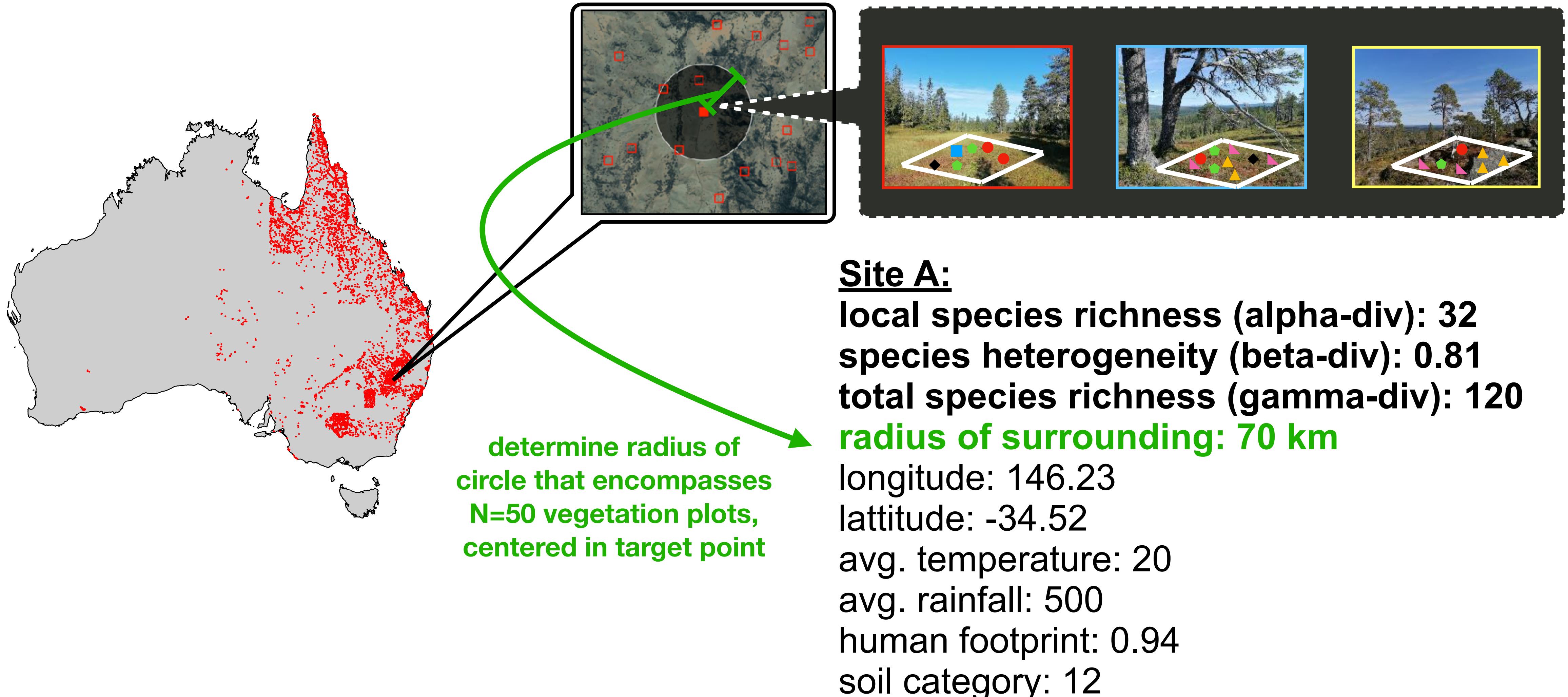
Preparing training data



determine radius of
circle that encompasses
N=50 vegetation plots,
centered in target point

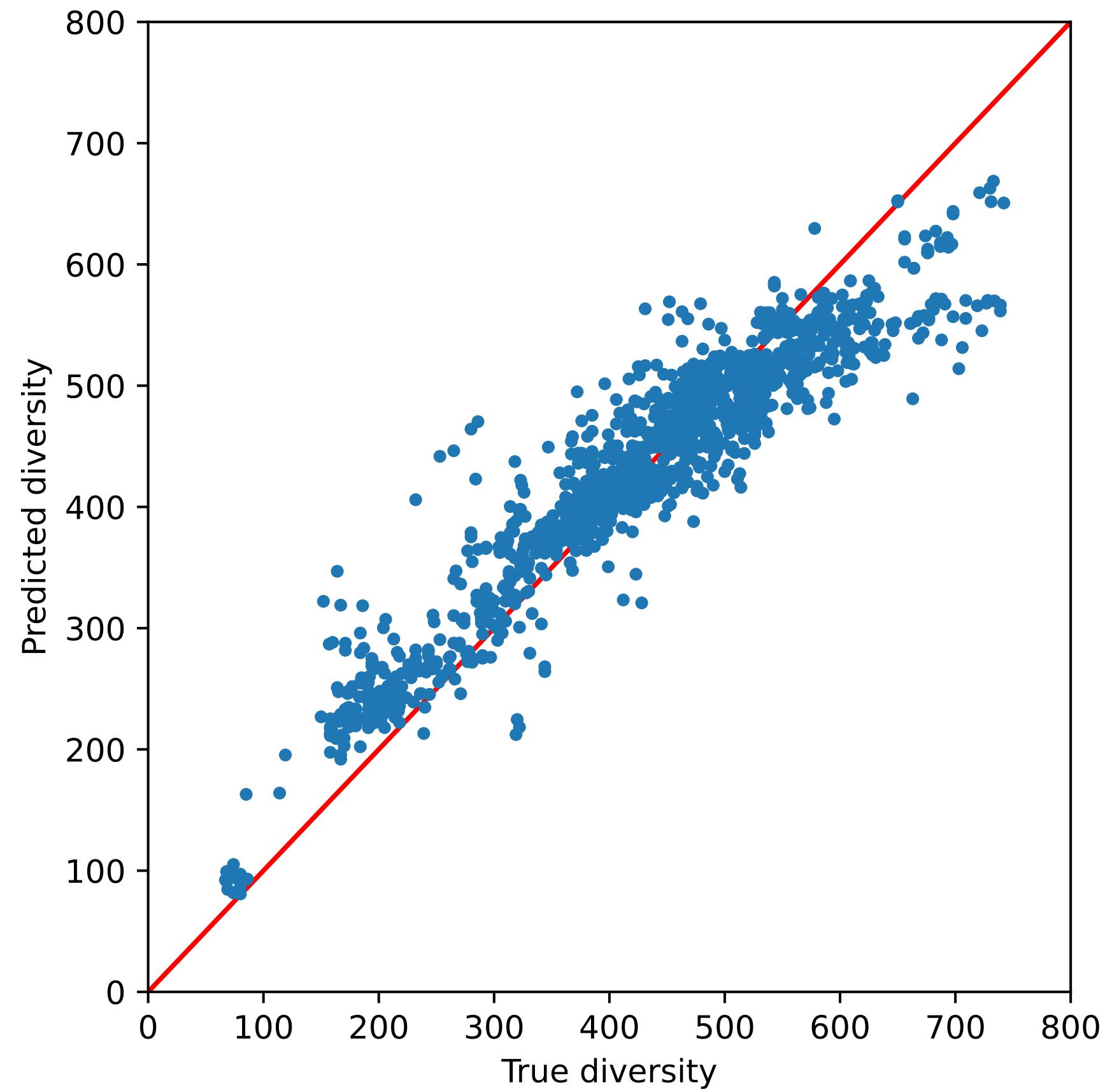
Andermann et al. (2022). *Frontiers in Plant Science*

Preparing training data

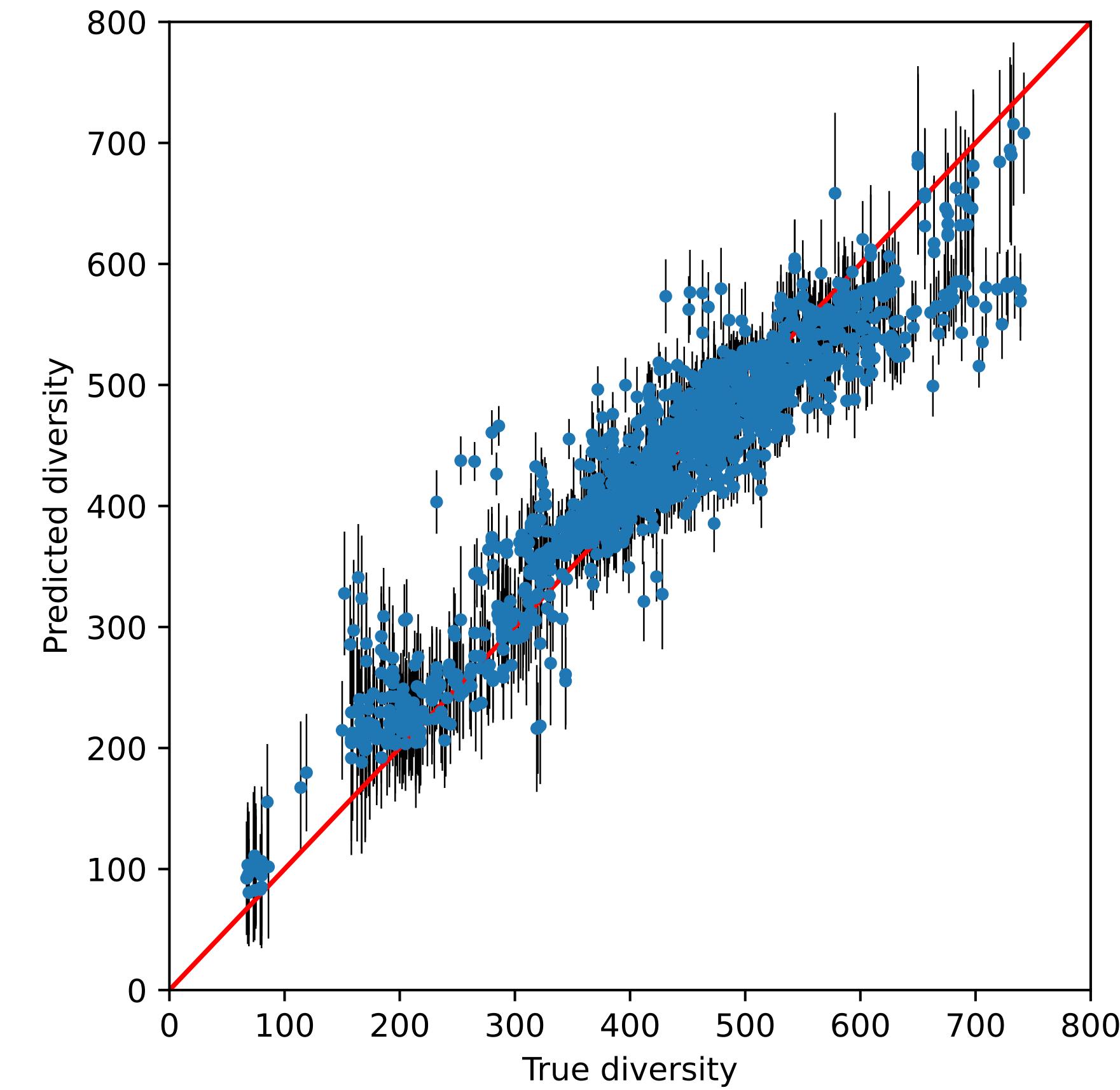


Andermann et al. (2022). *Frontiers in Plant Science*...

Quantifying uncertainty with MC dropout

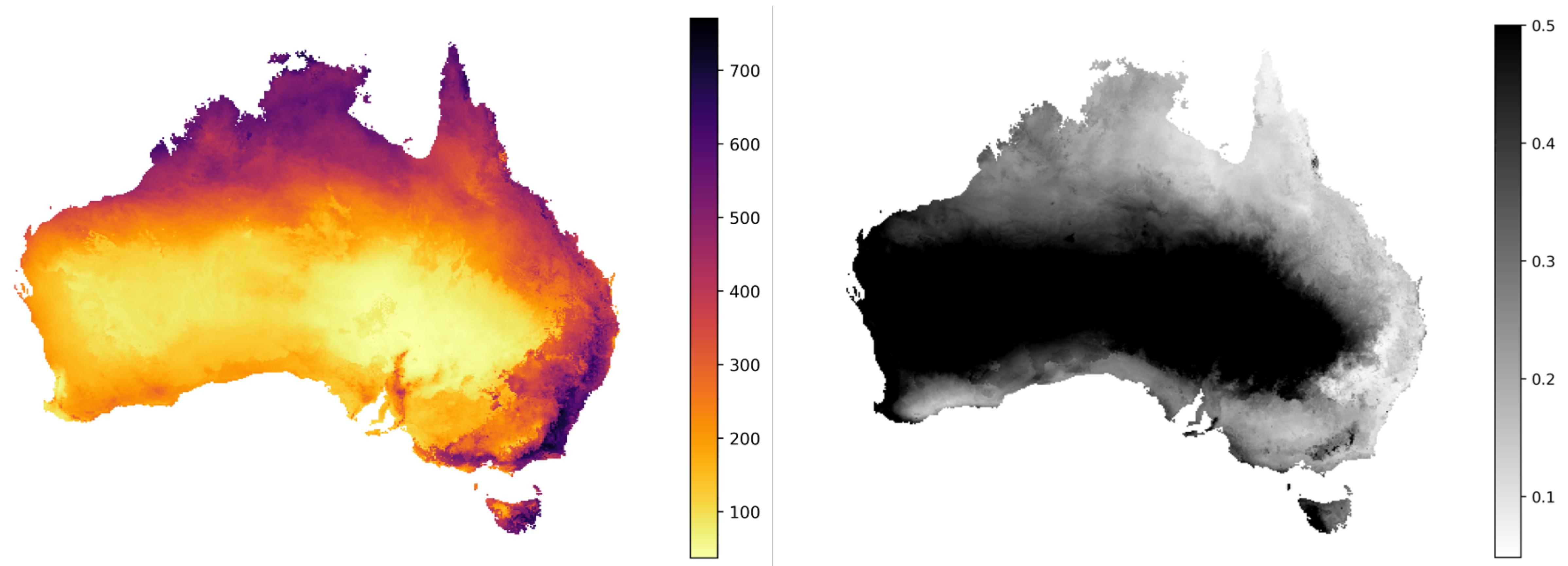


Test set predictions (single prediction)



MC dropout predictions (N=100)
estimating mean and standard deviation

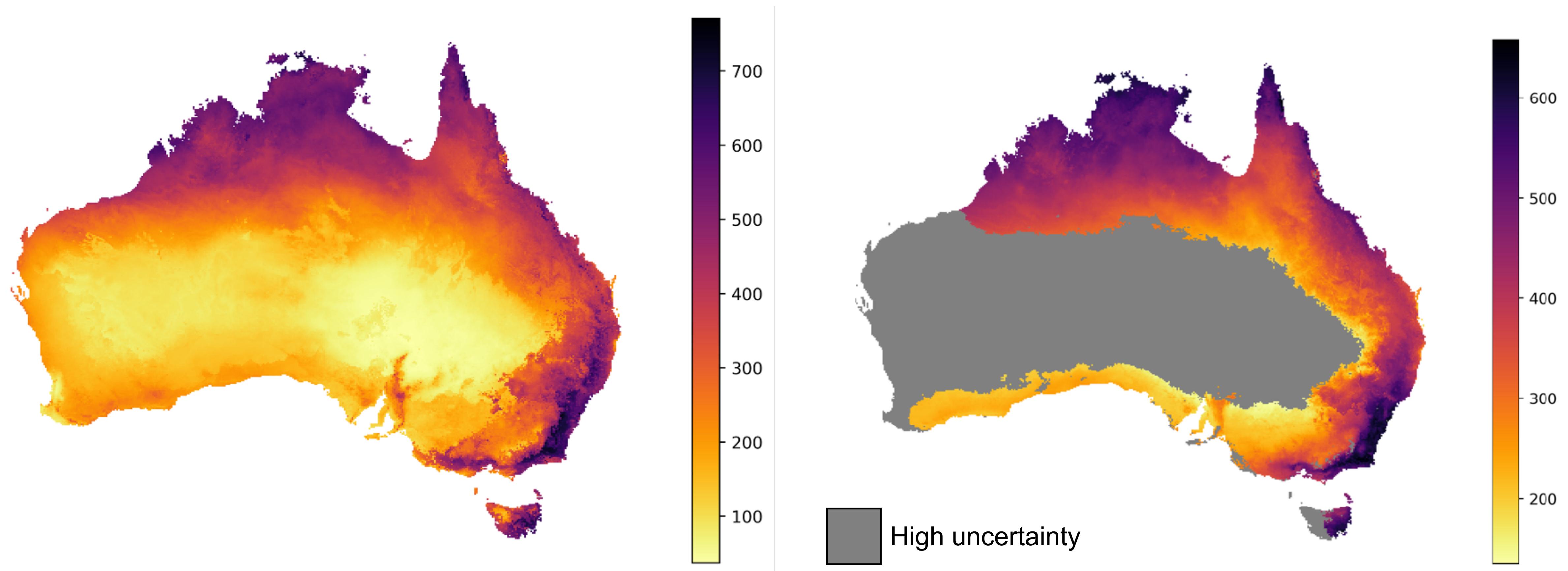
Quantifying uncertainty with MC dropout



Point estimate (single prediction)

Standard deviation of predictions across 100 MC dropout predictions

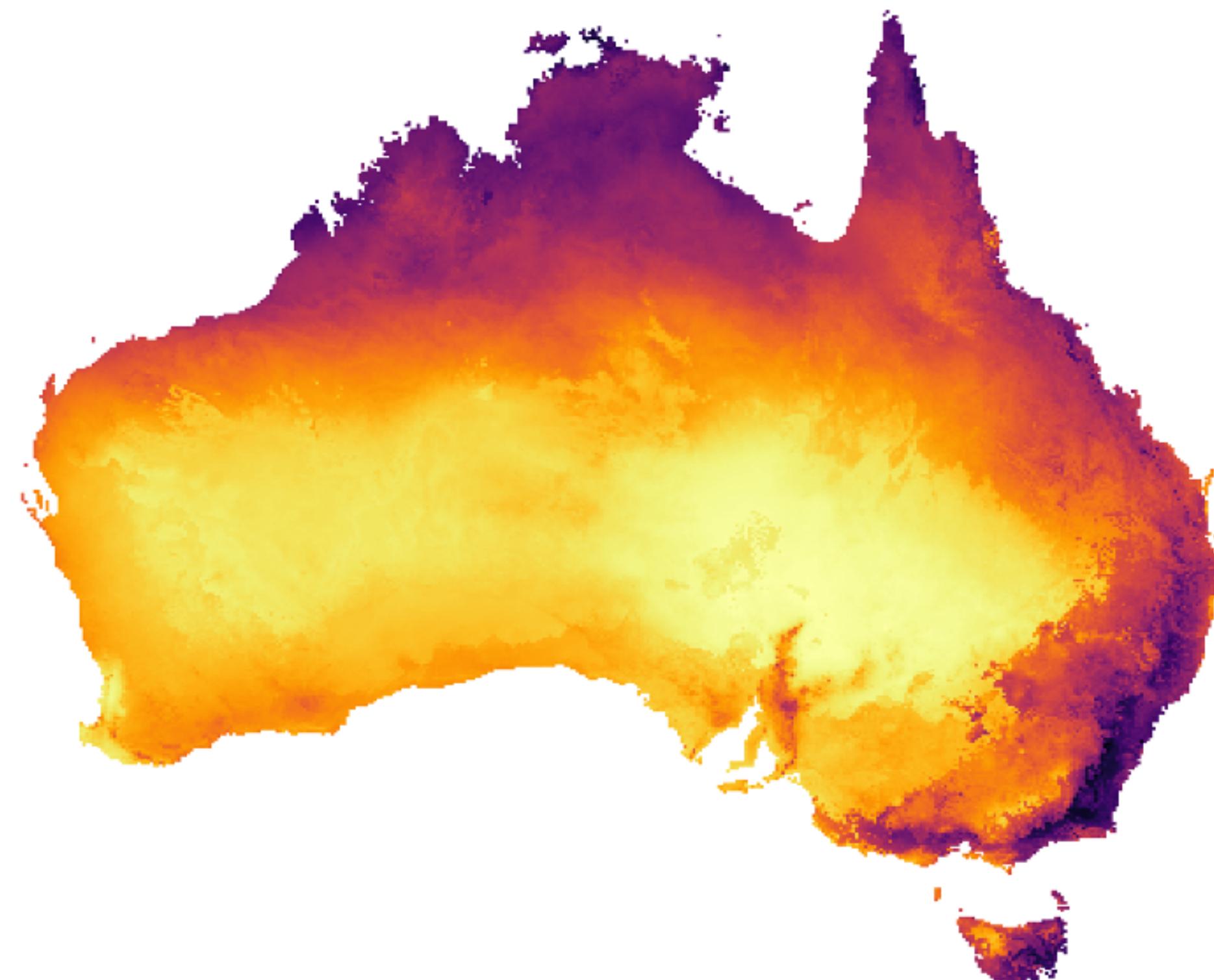
Quantifying uncertainty with MC dropout



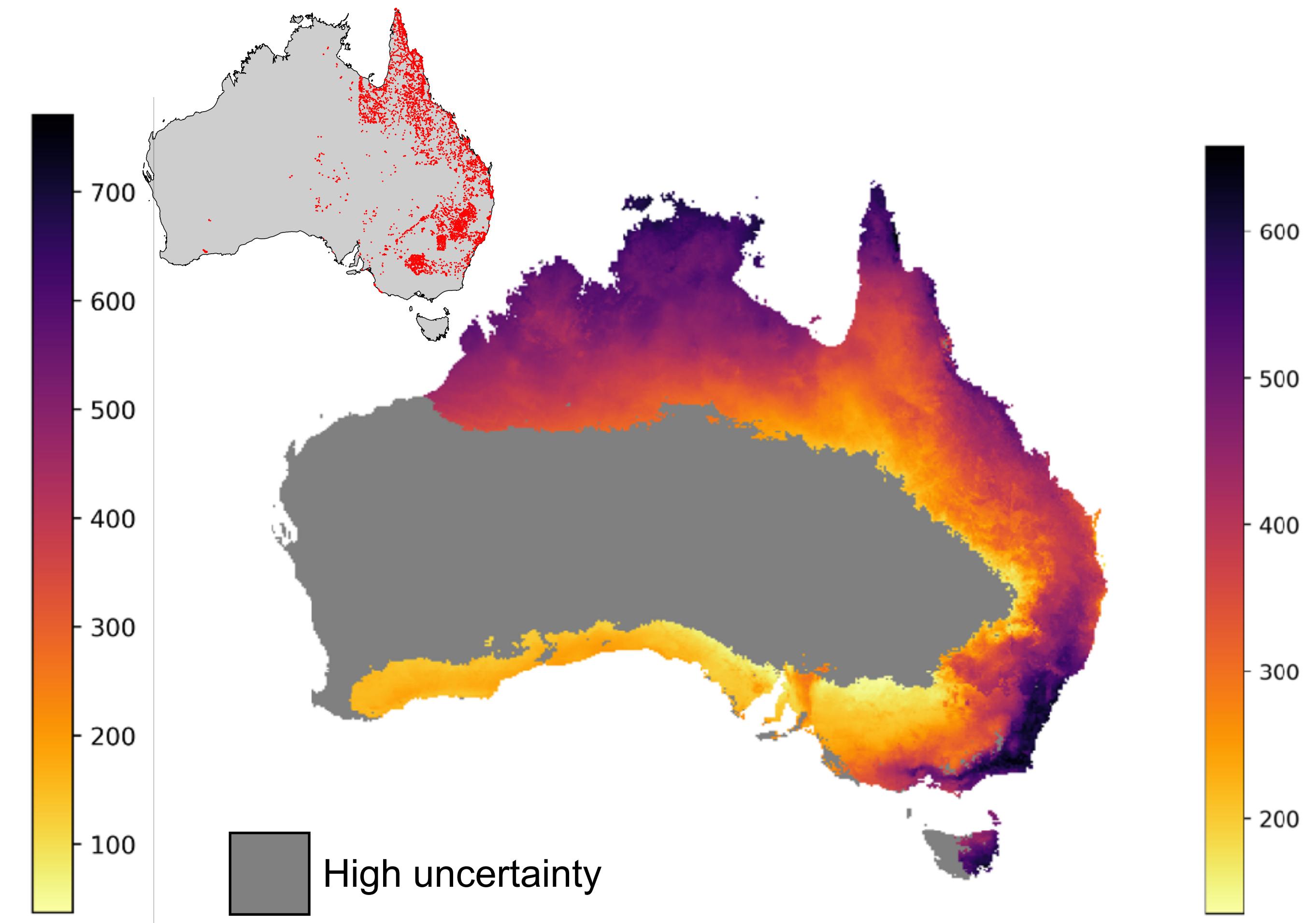
Point estimate (single prediction)

MC dropout predictions (N=100)
applying uncertainty cutoff (maximum standard deviation of 25%)

Quantifying uncertainty with MC dropout

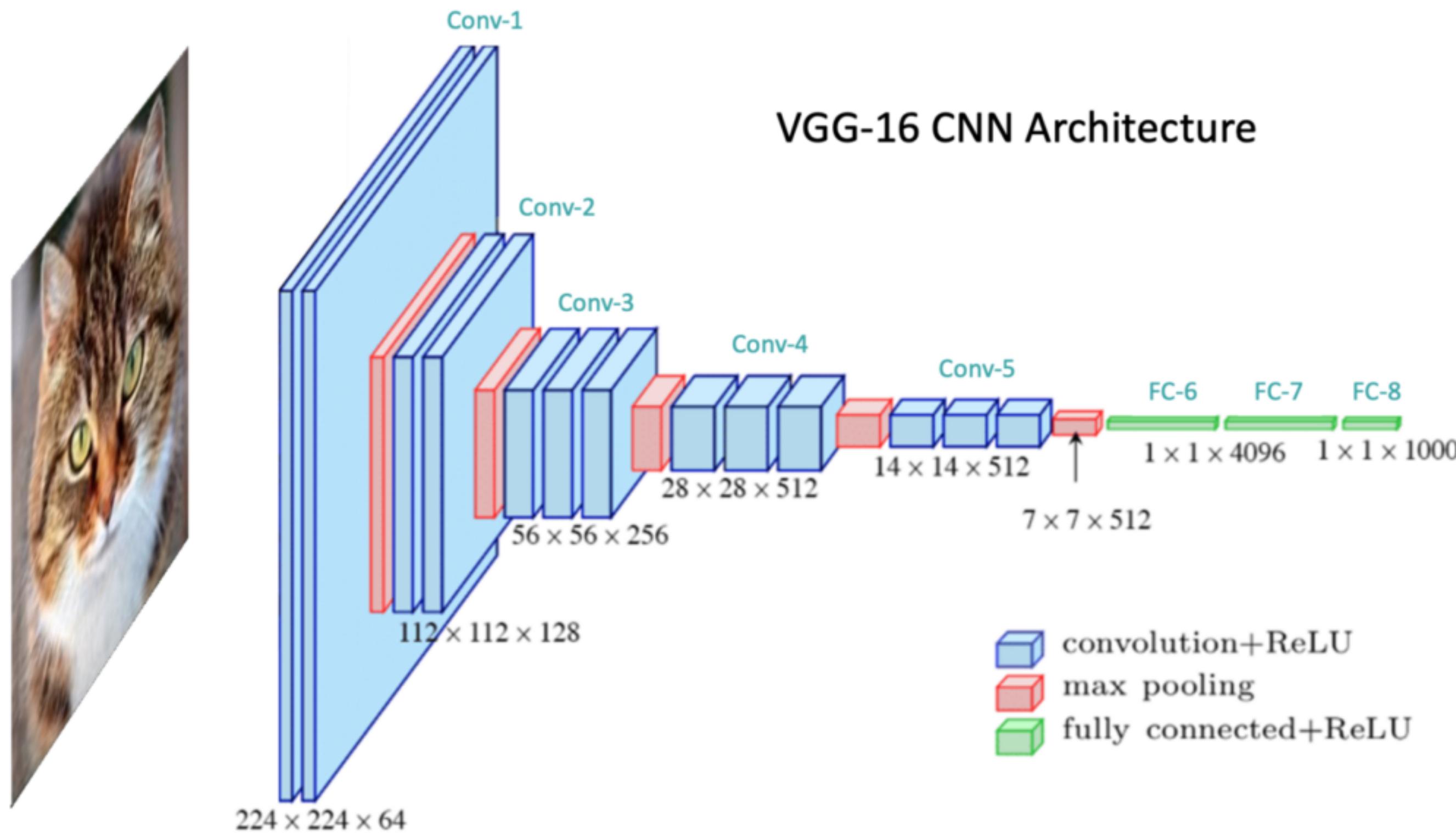


Point estimate (single prediction)

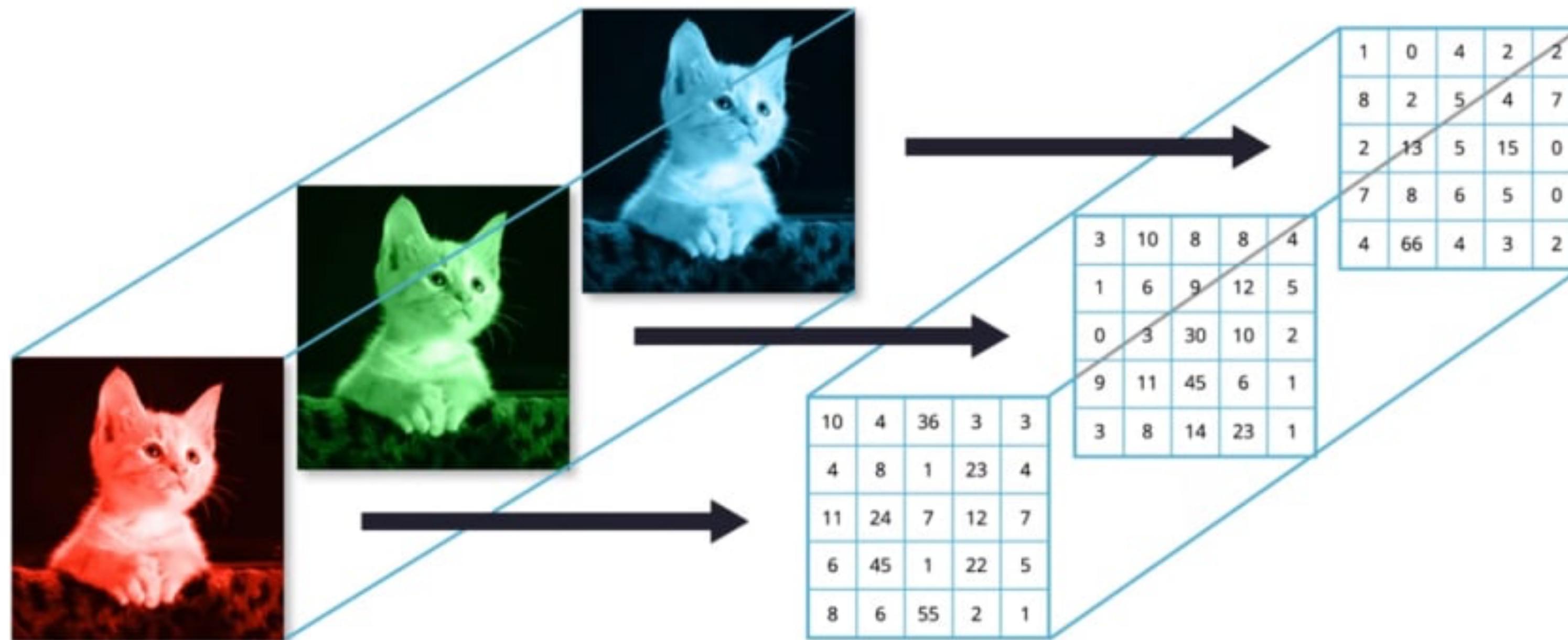


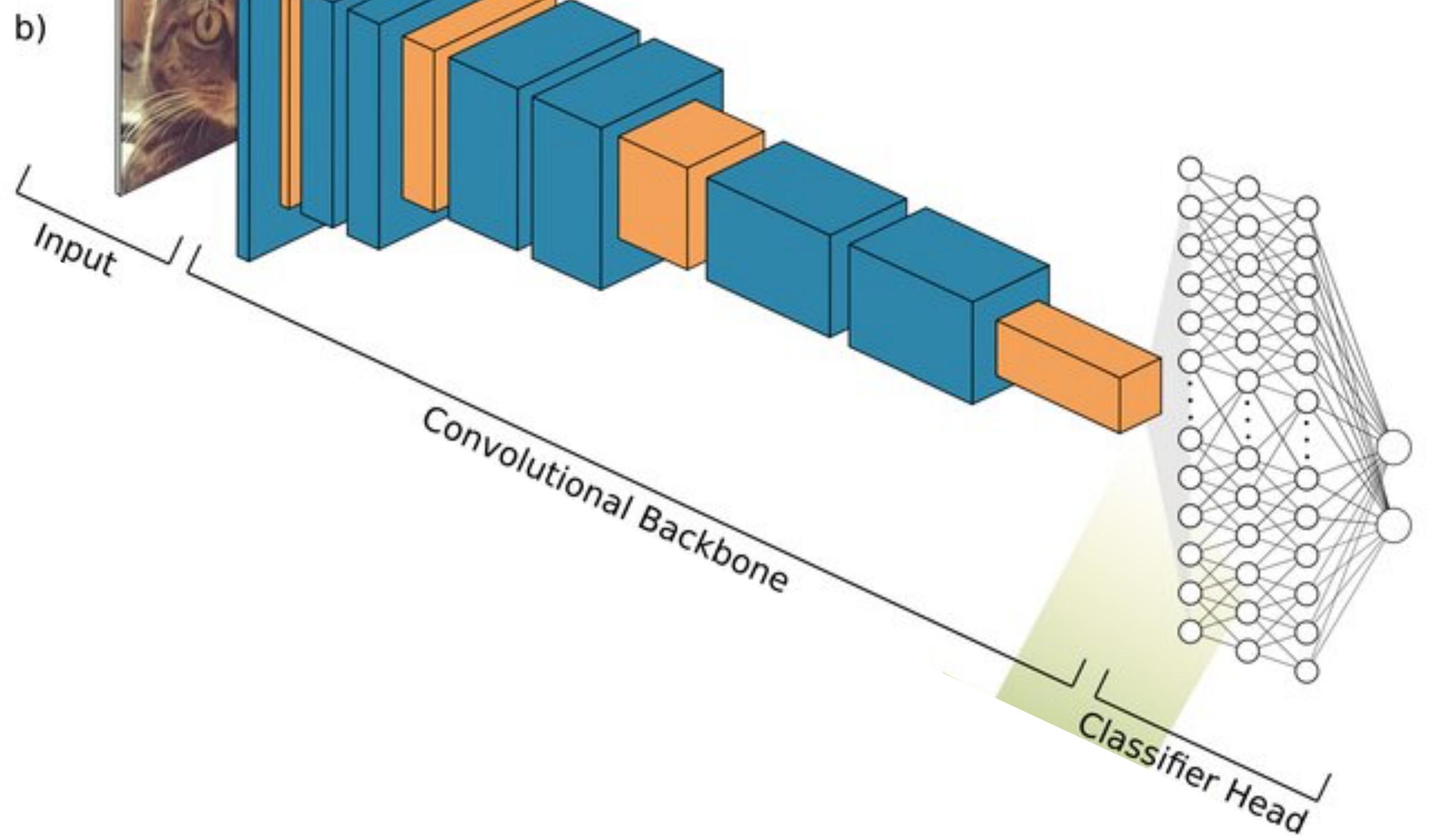
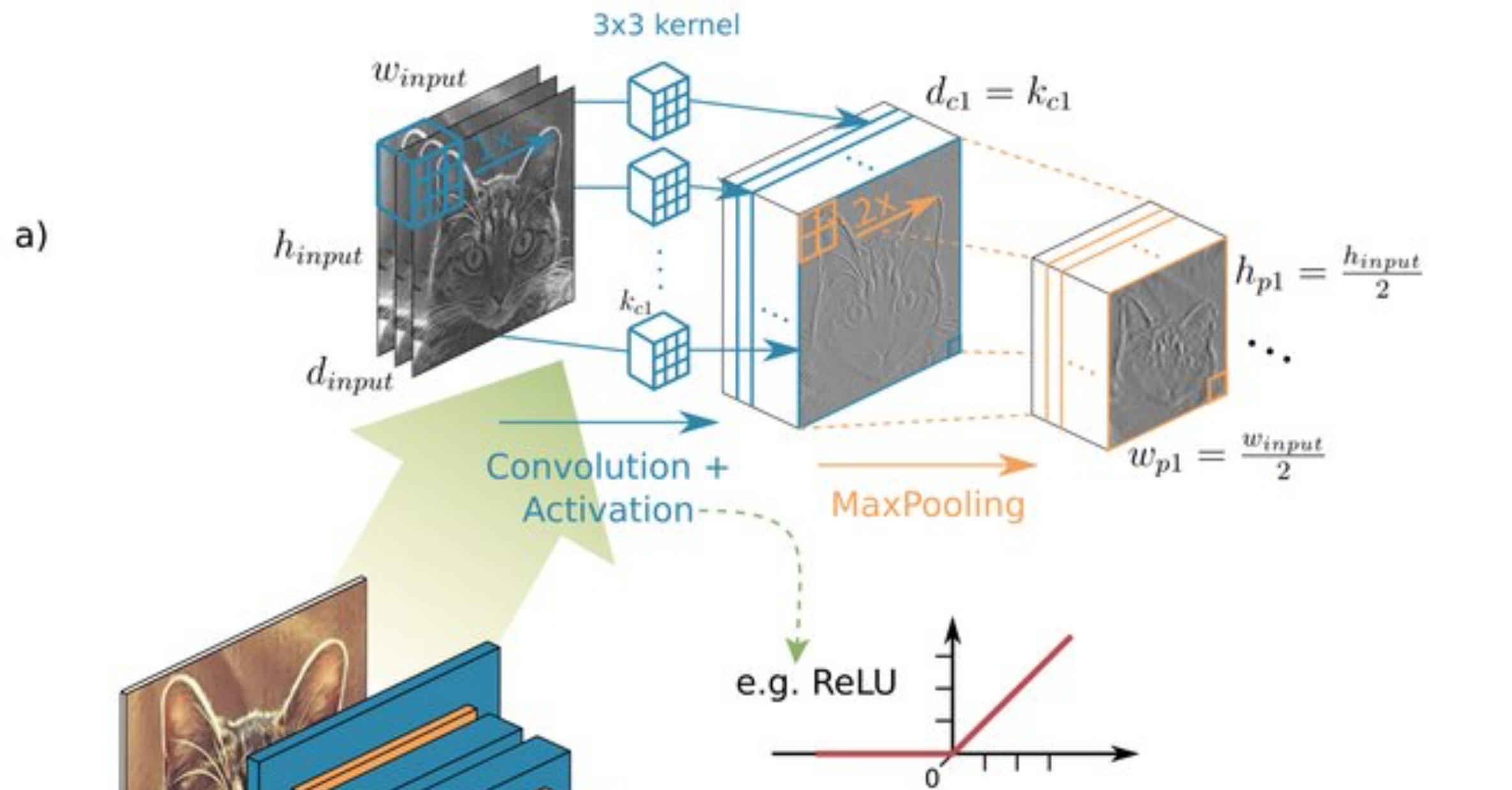
MC dropout predictions (N=100)
applying uncertainty cutoff (maximum standard deviation of 25%)

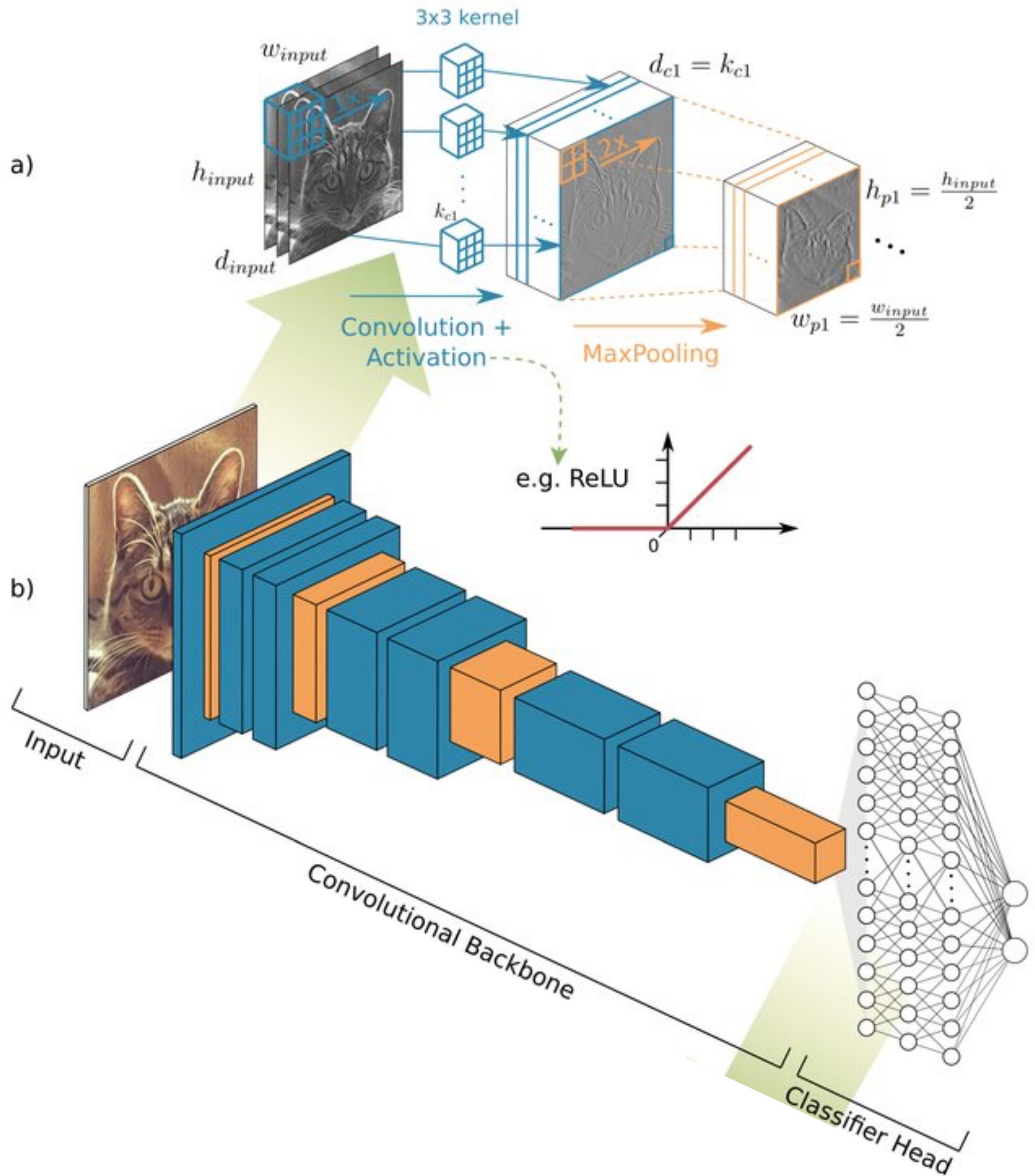
Convolutional neural networks



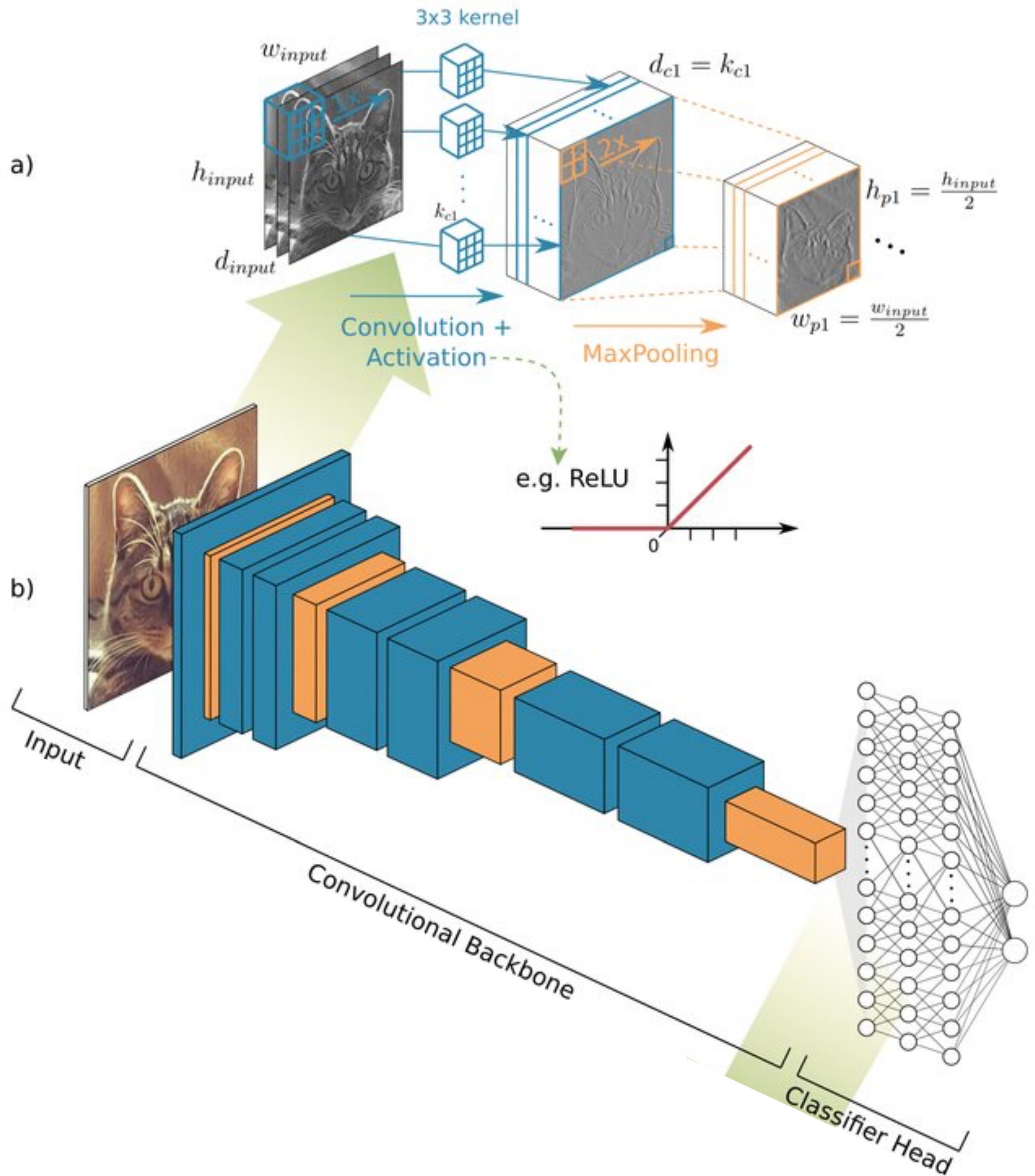
An image as data





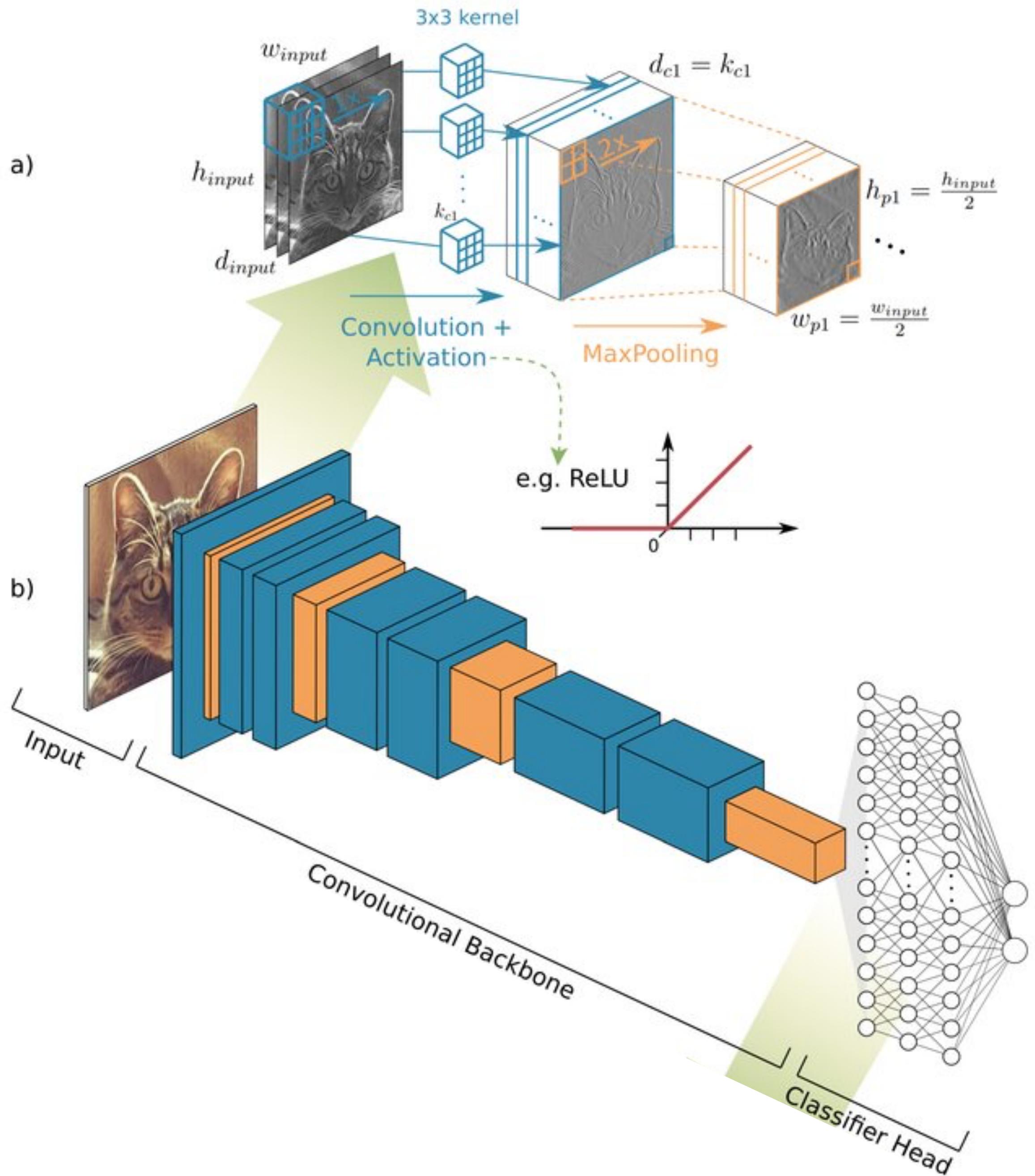


```
model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(64, 64, 3)),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10) # 10 different classes in the EuroSAT dataset
])
```



Number of kernels

```
model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(64, 64, 3)),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10) # 10 different classes in the EuroSAT dataset
])
```



Number of kernels

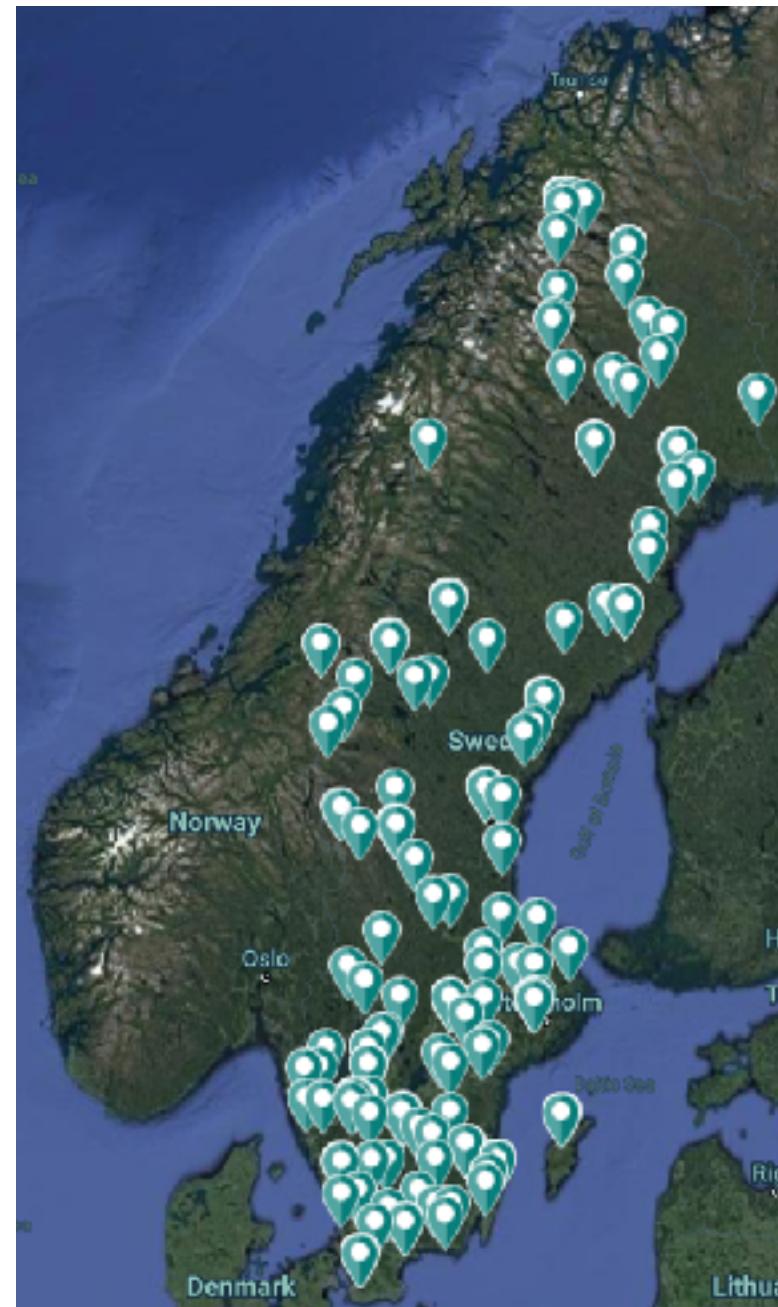
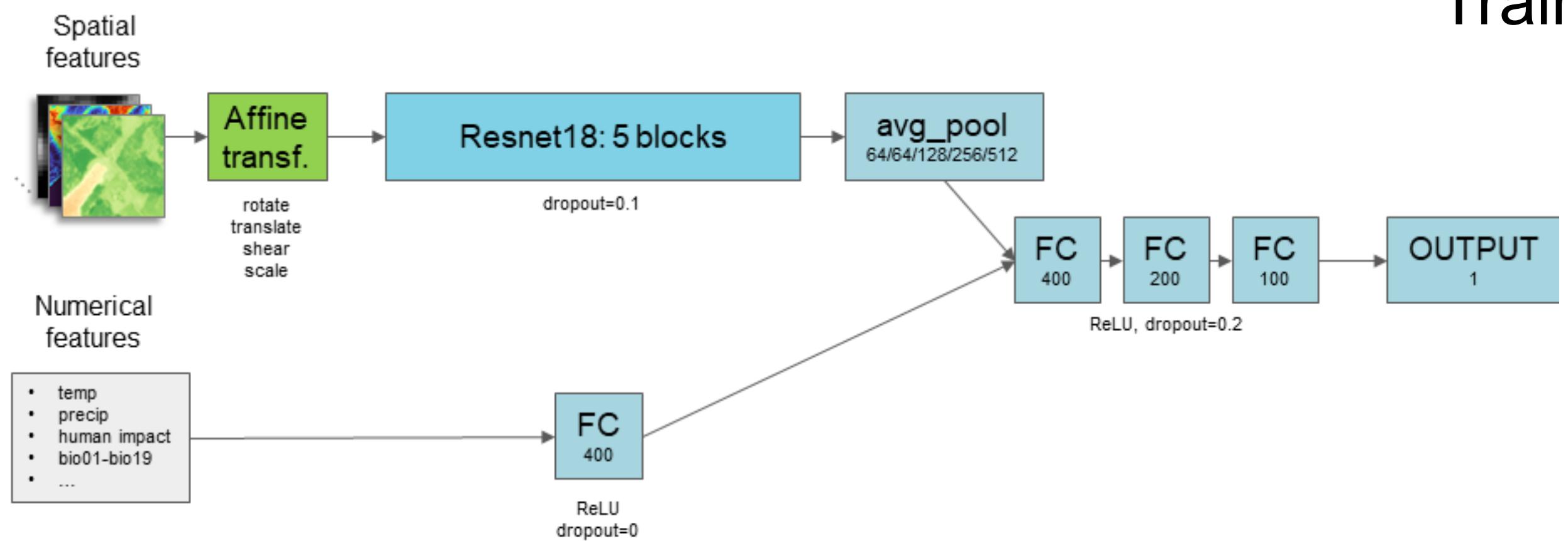
Size of kernels

```
model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(64, 64, 3)),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10) # 10 different classes in the EuroSAT dataset
])
```

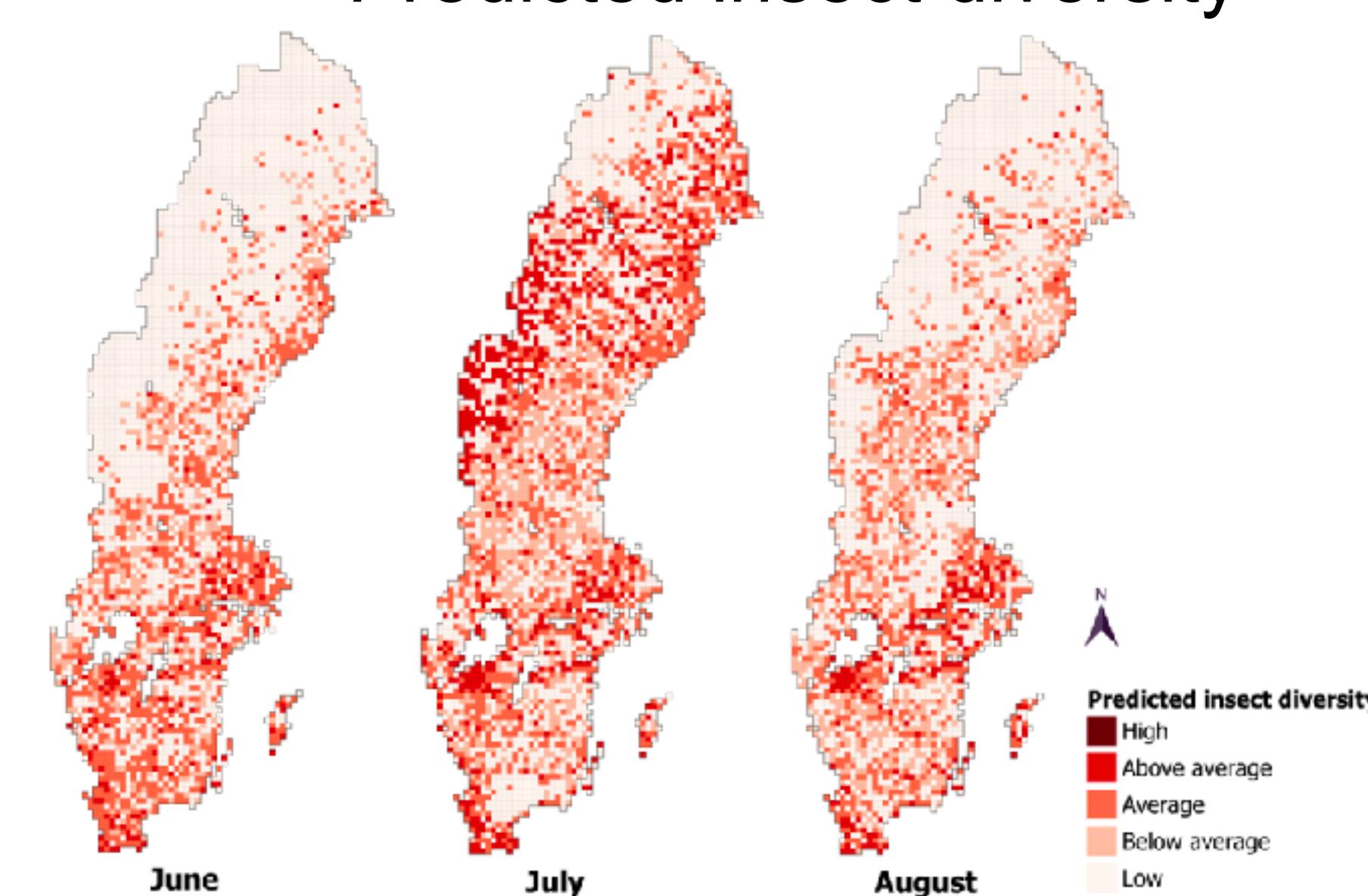
AI models trained on eDNA data



- Neural Network models trained on insect diversity from environmental DNA (eDNA) data
- Collaboration with Insect Biome Atlas, which collected eDNA data over several years on weekly basis
- 200 sites across Sweden



Training data

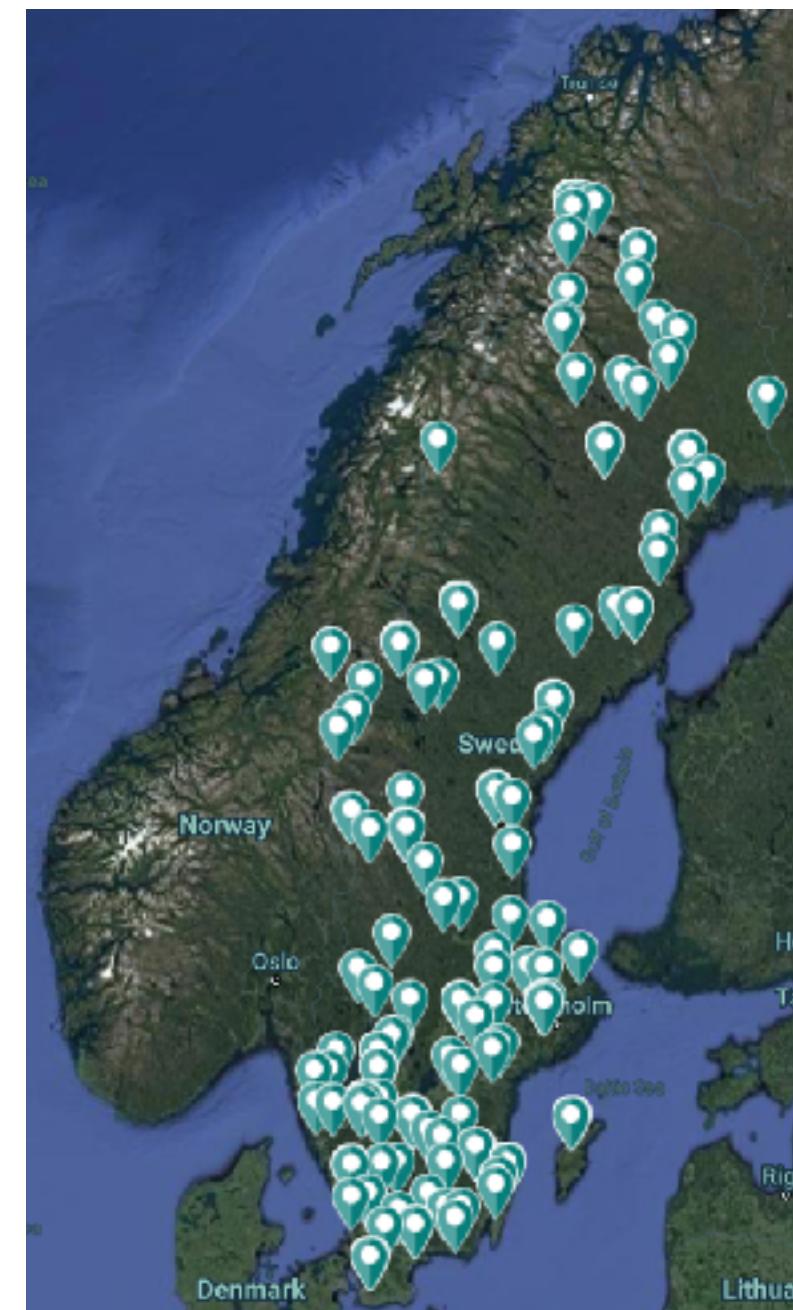
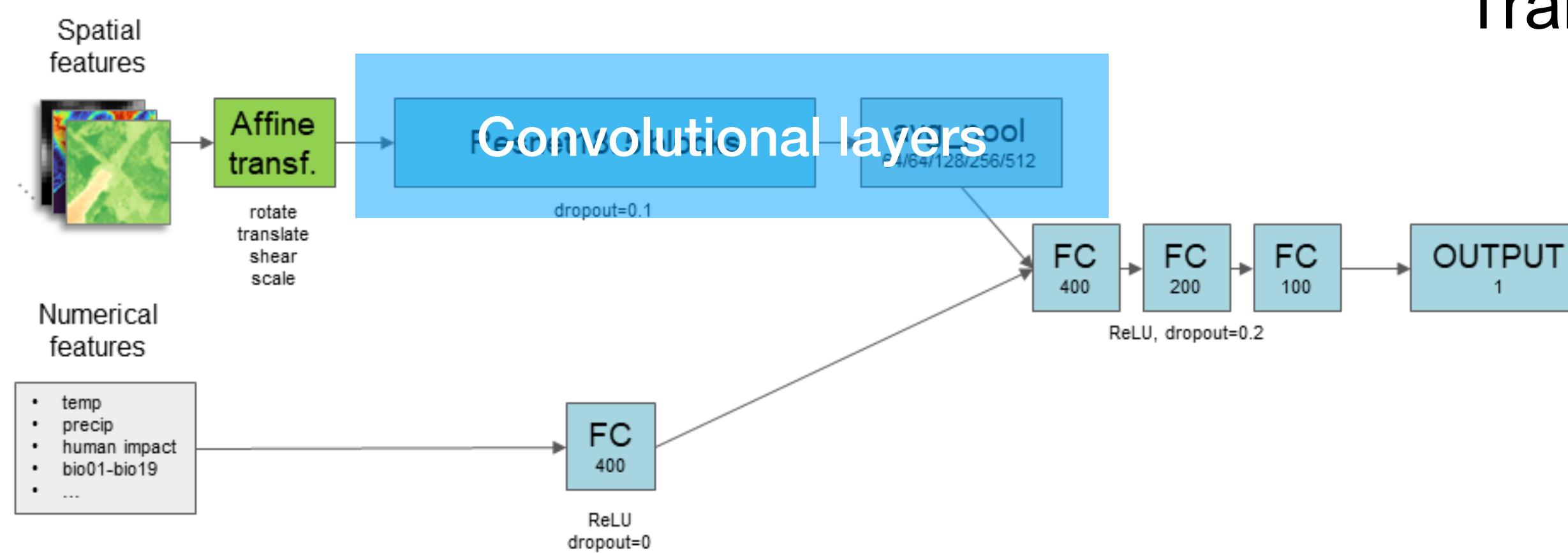


Baggström and Andermann, *in prep.*

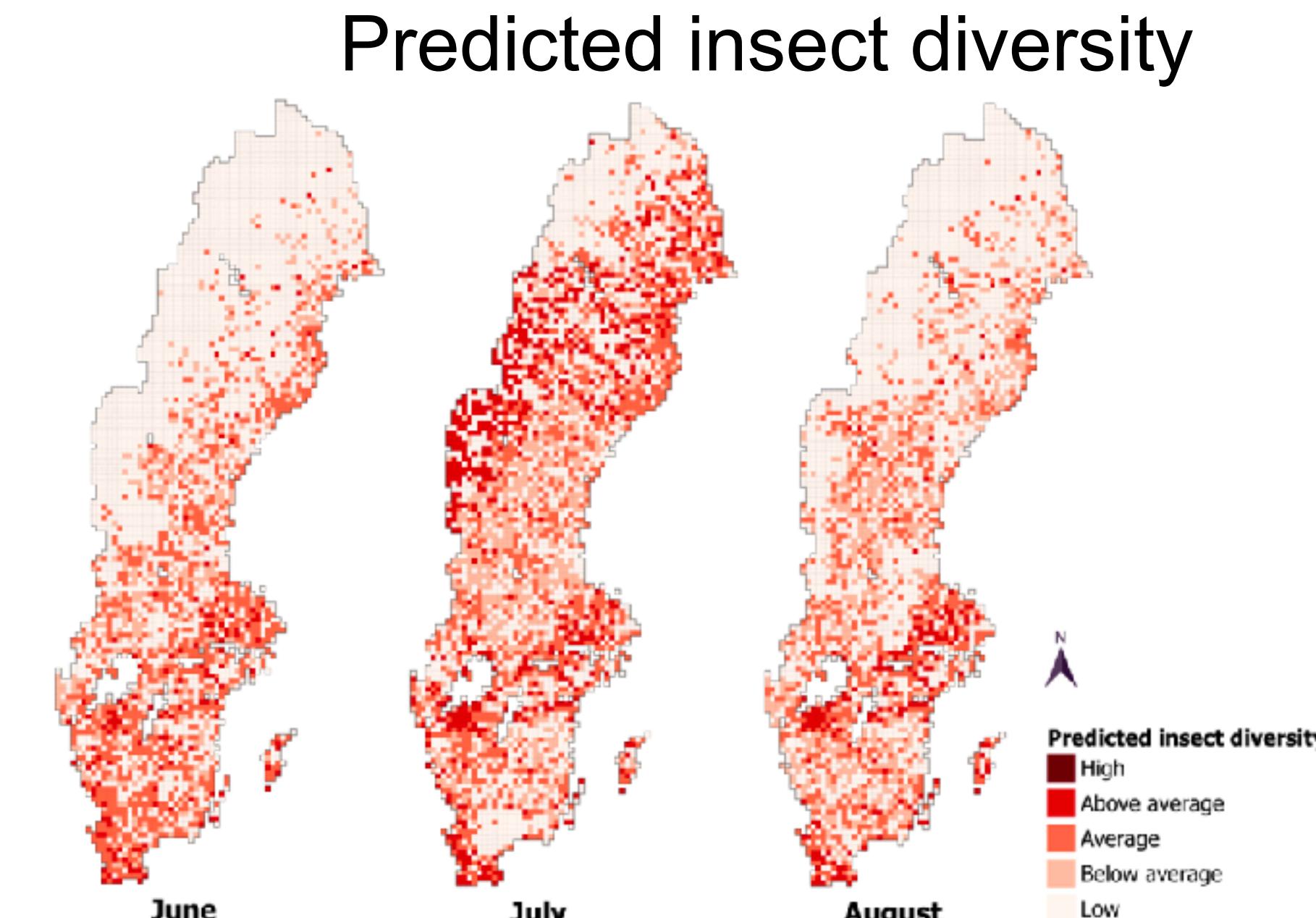
AI models trained on eDNA data



- Neural Network models trained on insect diversity from environmental DNA (eDNA) data
- Collaboration with Insect Biome Atlas, which collected eDNA data over several years on weekly basis
- 200 sites across Sweden



Training data



Baggström and Andermann, *in prep.*