
Using R to build artificial neural networks in medical data

Thomas Wollseifen

13-Nov-2019

Agenda

1. Introduction to Artificial Neural Networks (ANN)
2. ANNs with R - `neuralnet`
3. Classifying Diabetes with ANNs
4. TensorFlow and Convolutional Neural Networks (CNN) with R - `keras`
5. Summary

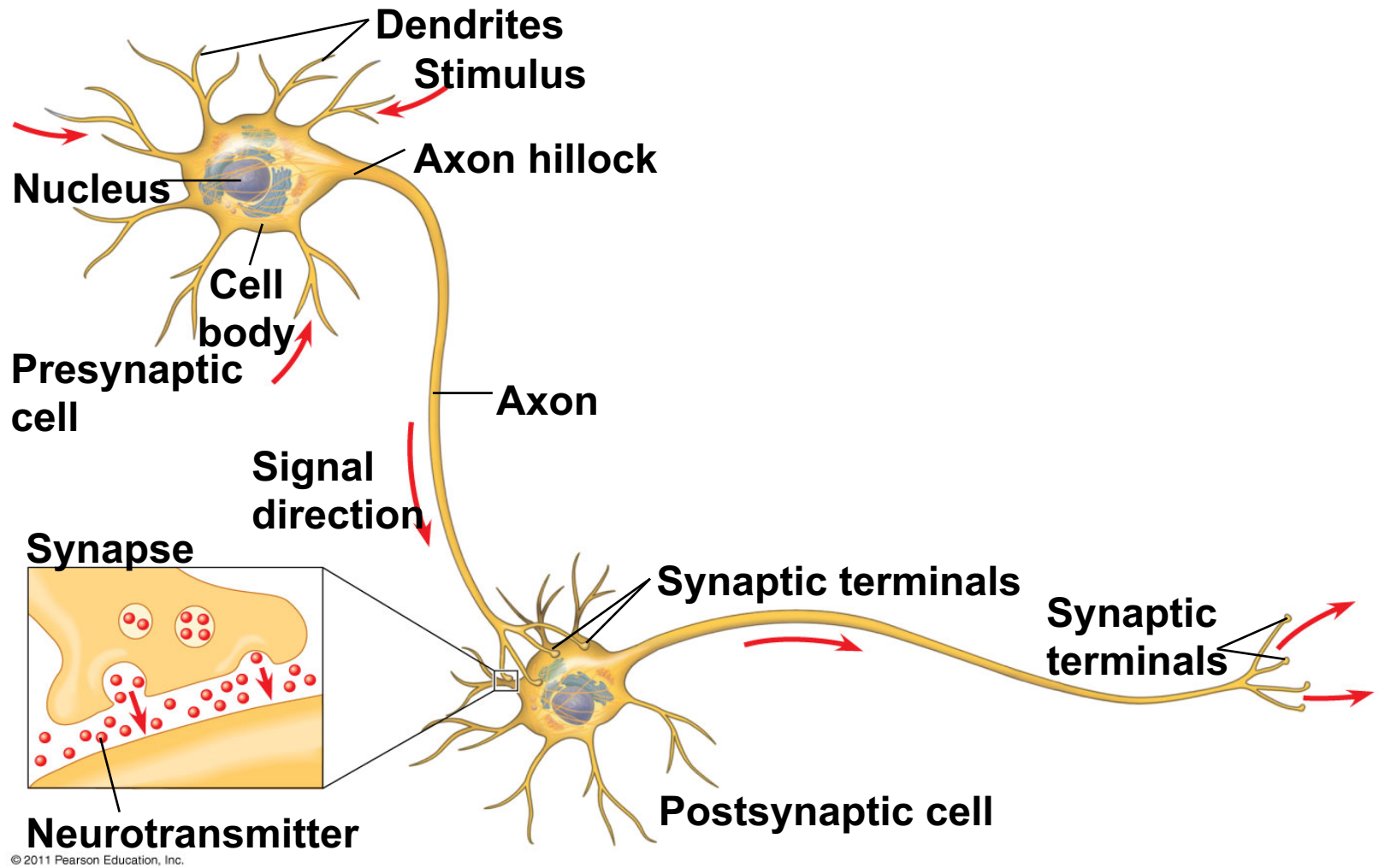
Introduction to Artificial Neural Networks

$\sim 10^{11}$ neurons
in human brain

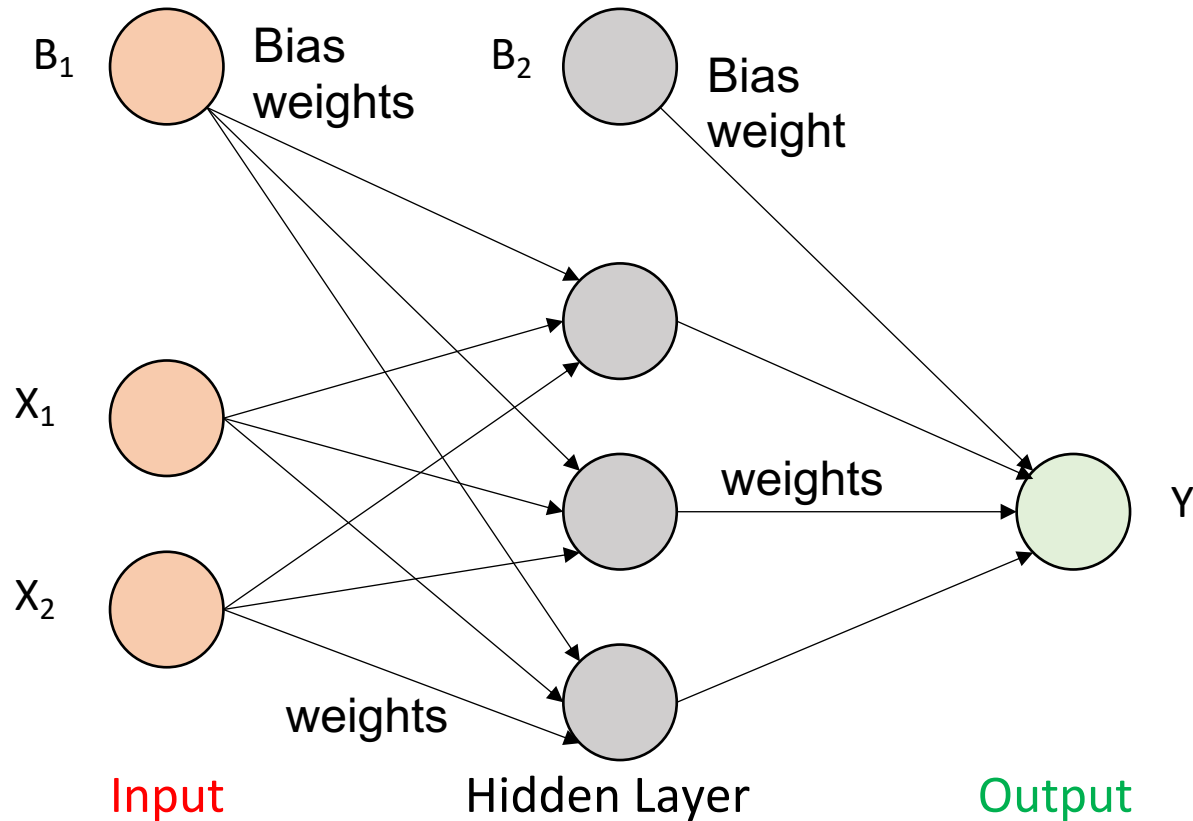
$\sim 10^4$ connections per neuron

$\sim 10^{15}$ connections

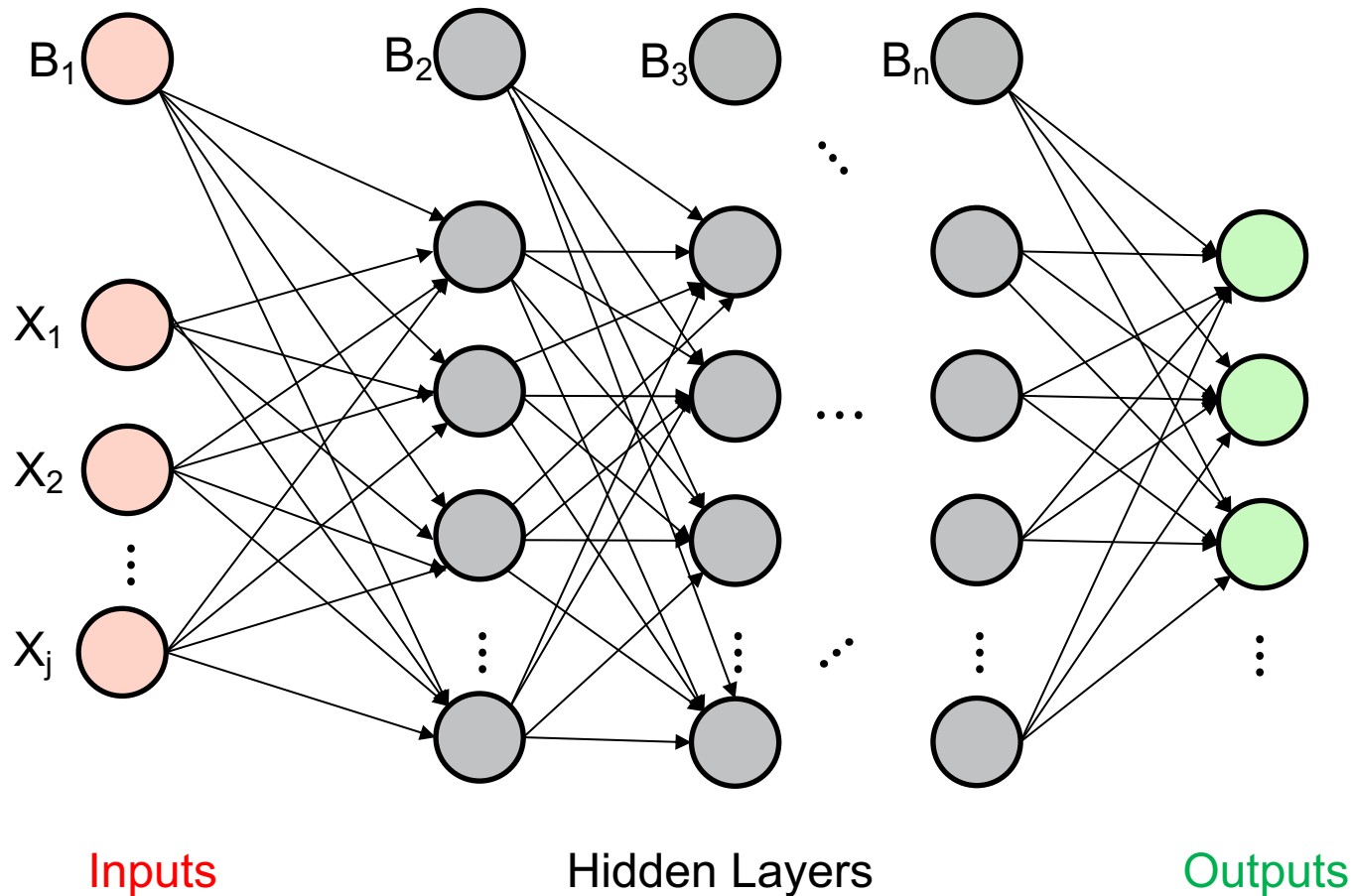
Neuron - Signal flow from dendrites to axon ends



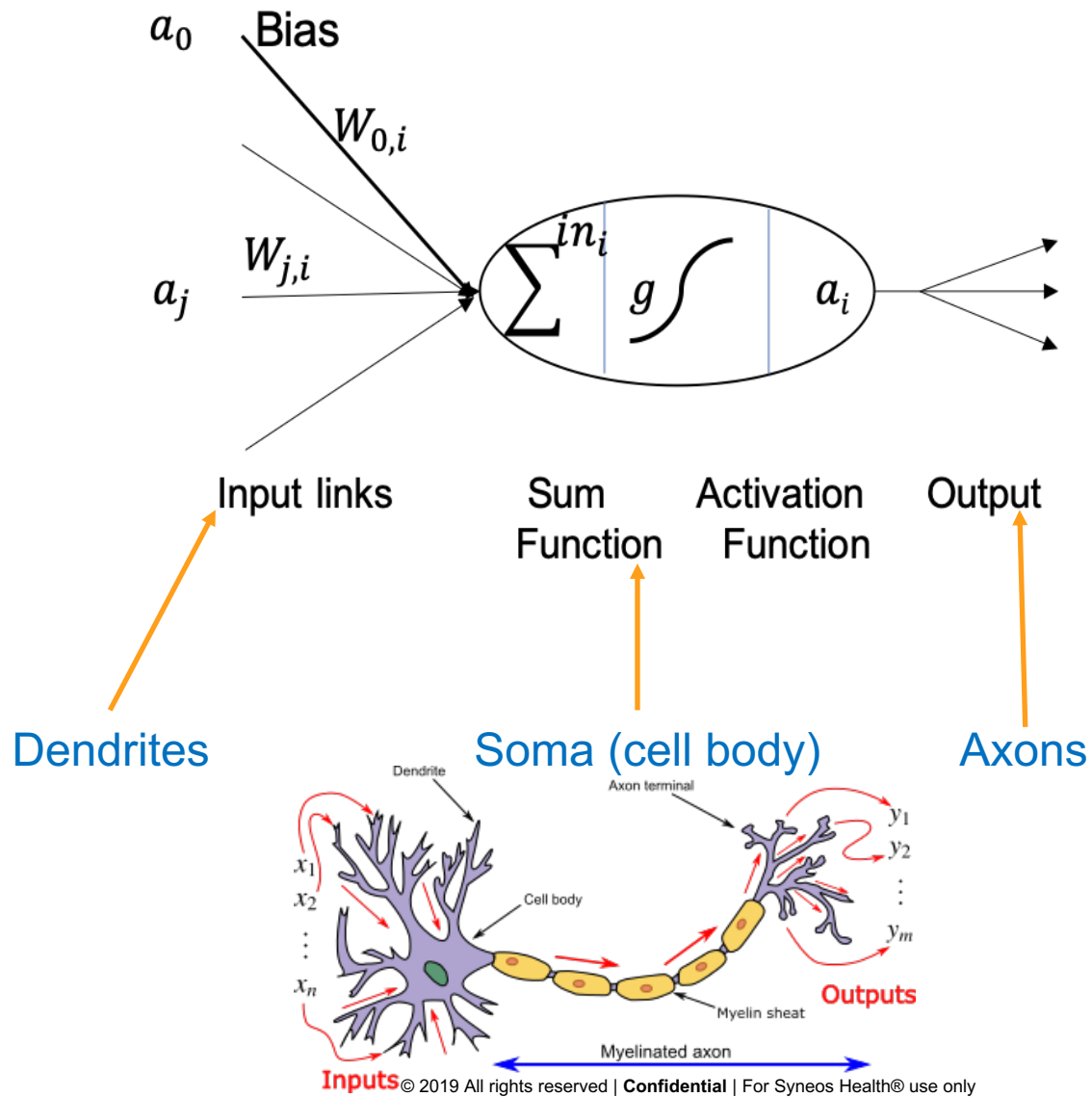
Artificial Neural Network – 1 hidden layer



Artificial Neural Network - multilayers



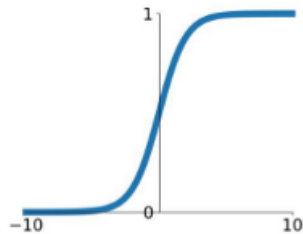
Model of a neuron



Activation functions

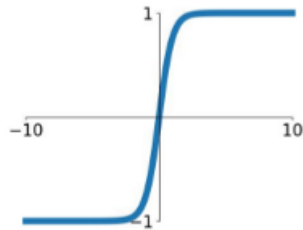
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



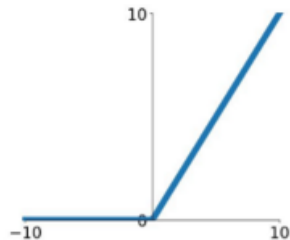
tanh

$$\tanh(x)$$



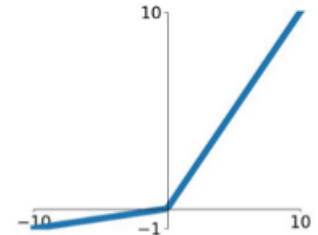
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

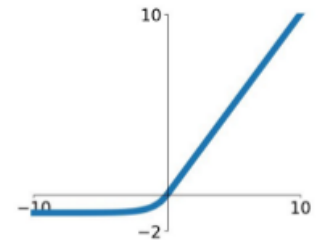


Maxout

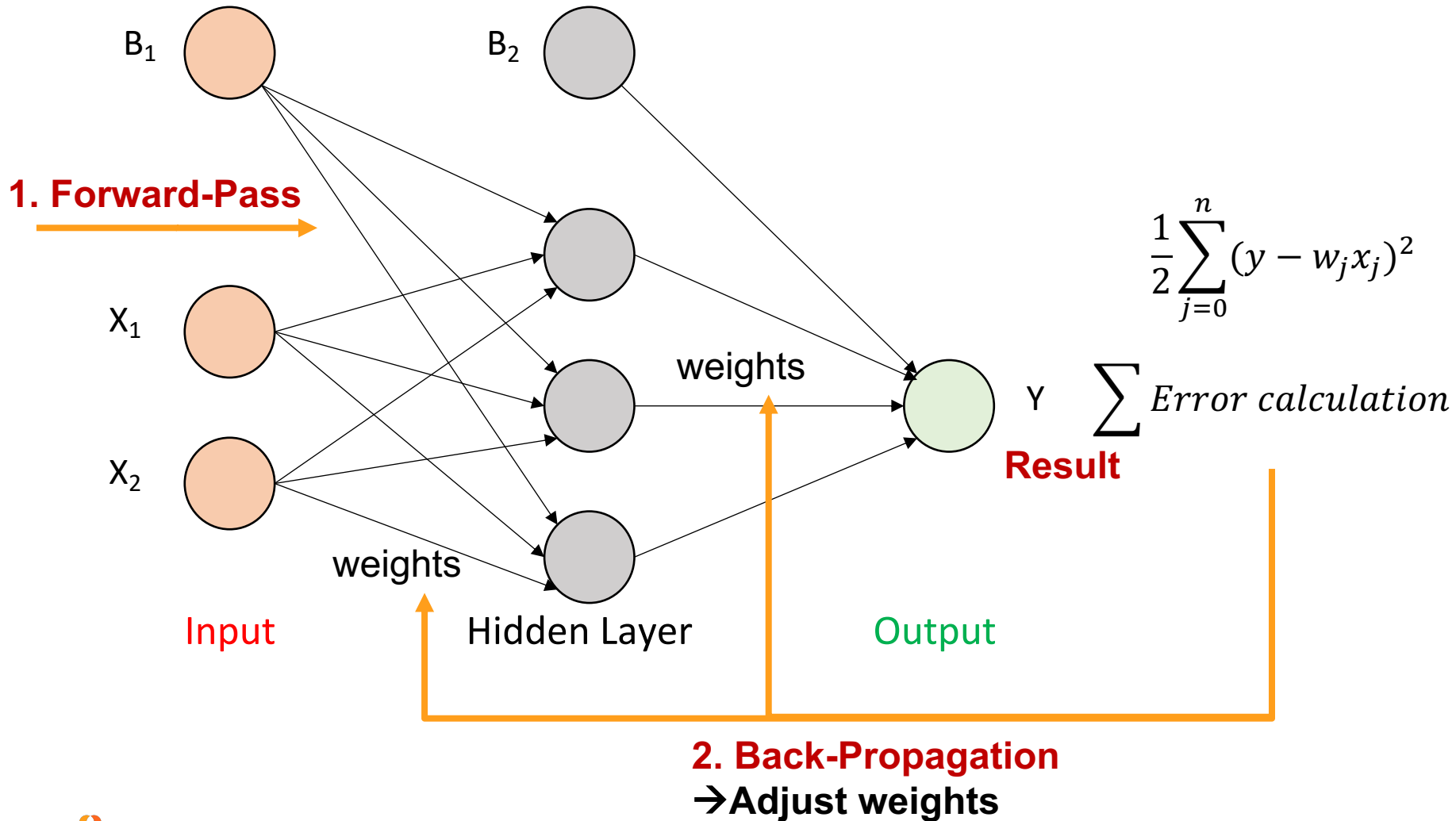
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

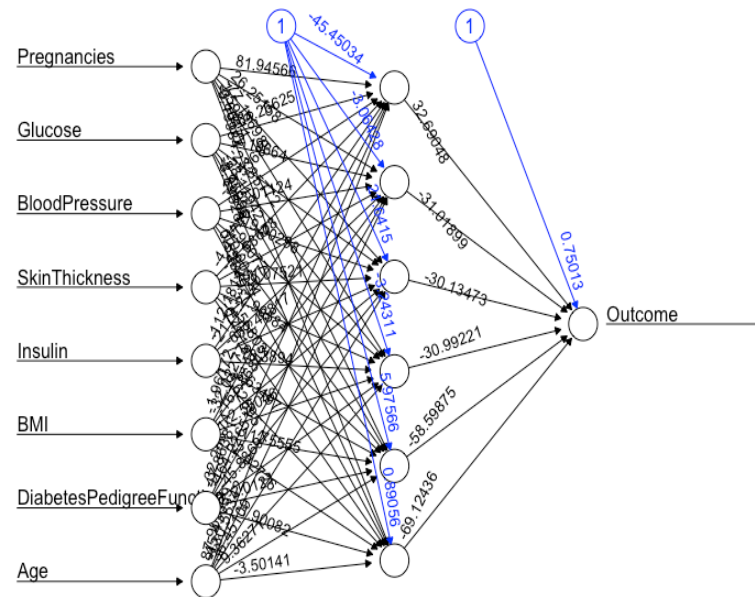


Feed-Forward Pass / Back-Propagation Algorithm



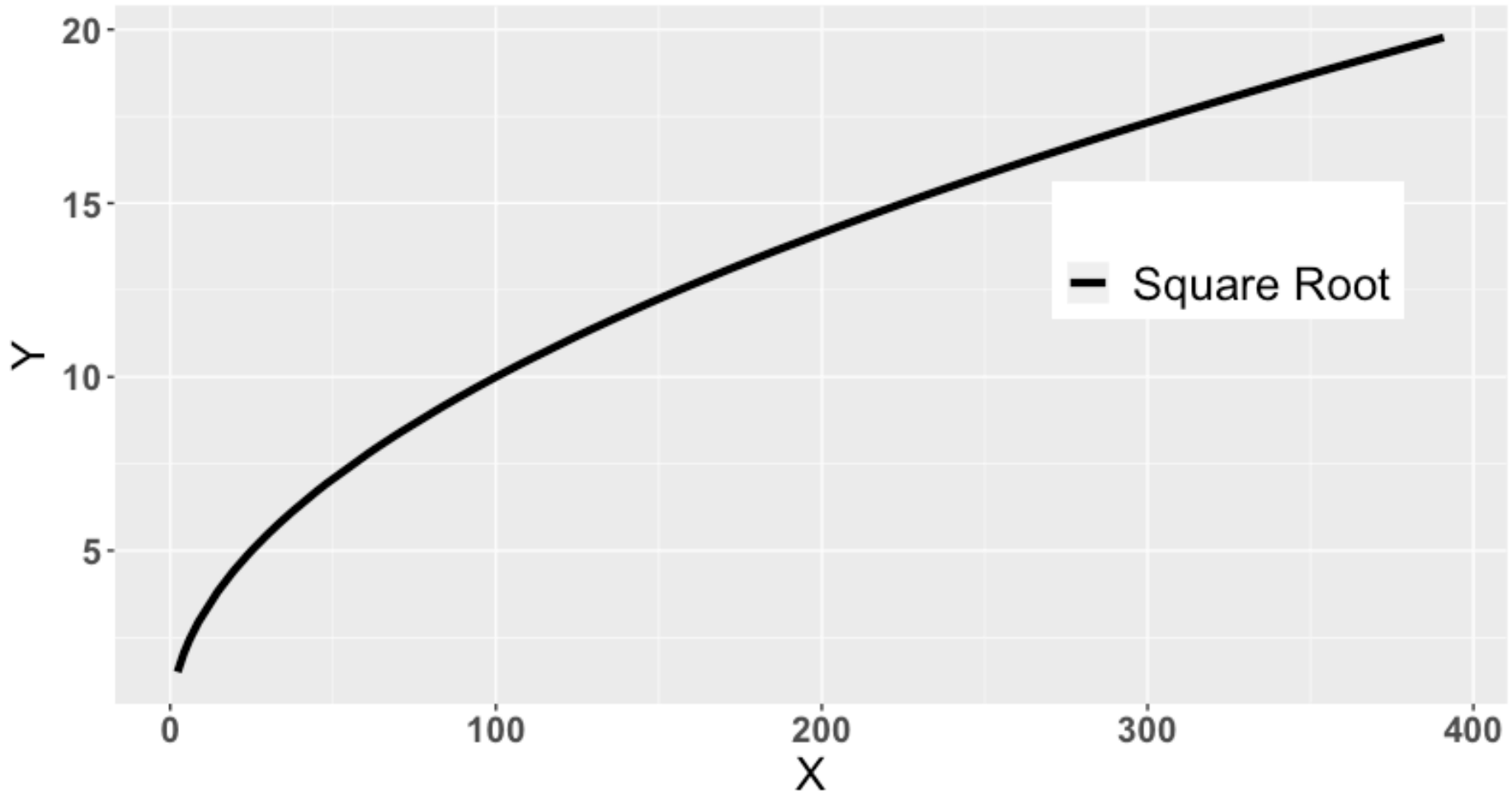
Artificial Neural Networks with R

Package neuralnet



Error: 37.668814 Steps: 44984

Example 1 – Square Root Function

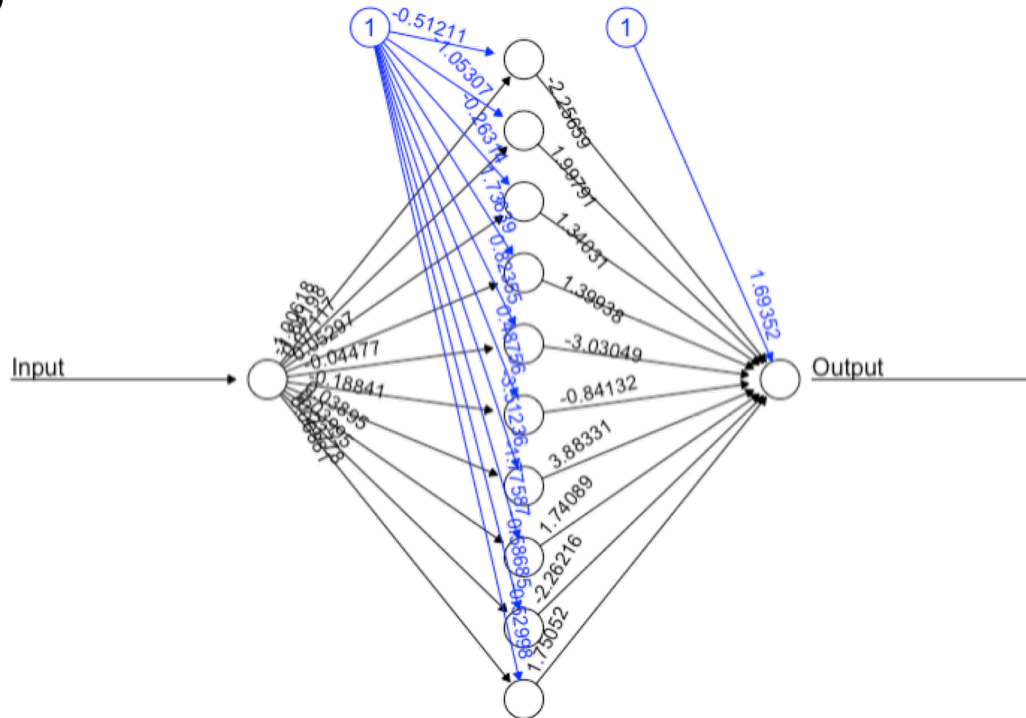


Build the ANN with Training Data

```
traininginput <- as.data.frame(runif(50, min=0, max=200))  
trainingoutput <- sqrt(traininginput)
```

```
net.sqrt <-neuralnet(Output~Input,trainingdata,hidden=10,threshold=0.01)
```

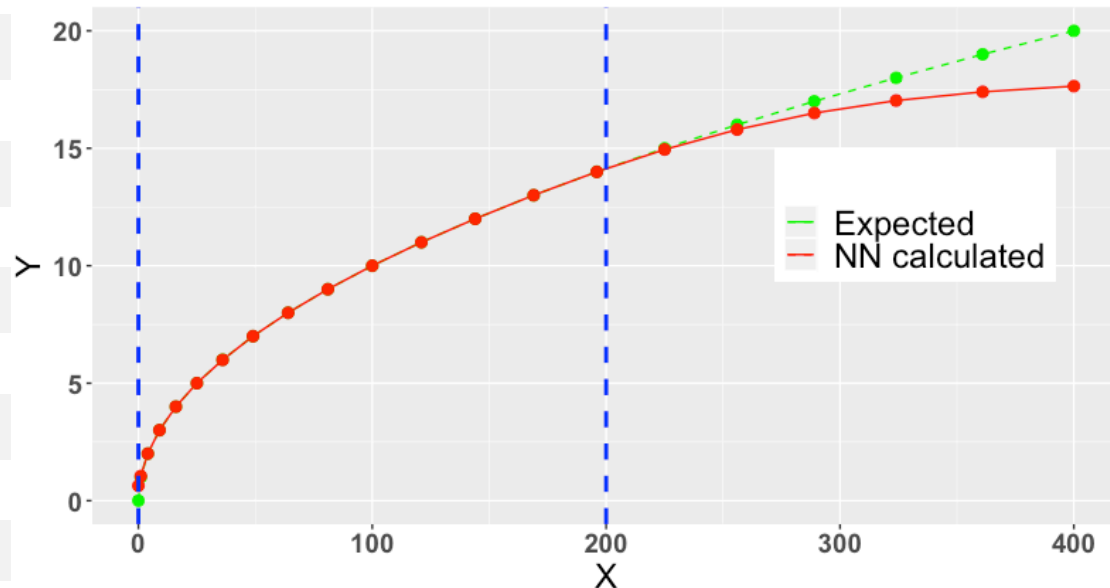
```
plot(net.sqrt)
```



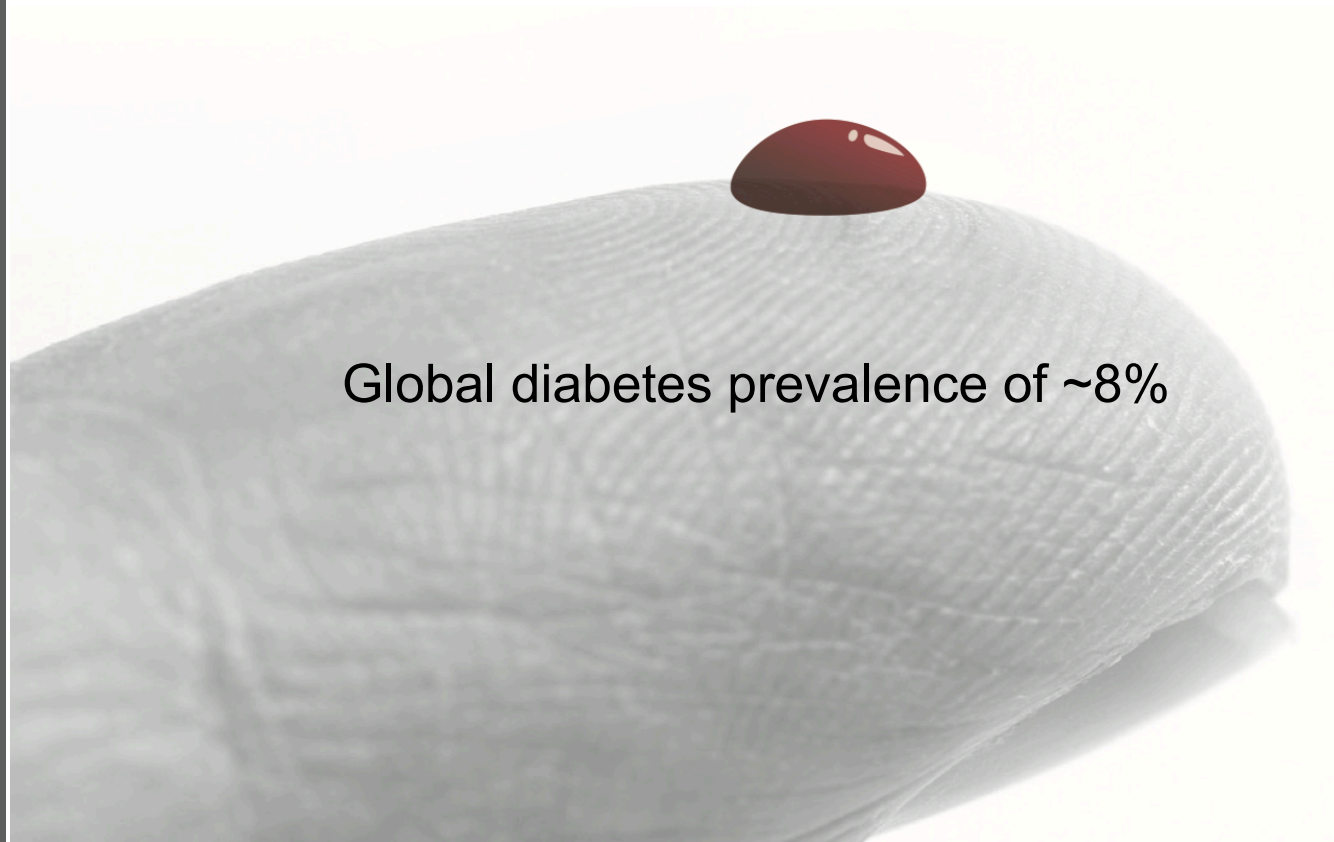
Test the ANN

```
testdata <- as.data.frame((0:20)^2)
net.results <- predict(net.sqrt, testdata)
results<-as.data.frame(net.results)
```

Input	Square root expected	Neural net output predicted
0	0	0.7143873
1	1	1.0243201
4	2	2.0041862
9	3	3.0001876
16	4	4.0006444
25	5	4.9958750
36	6	6.0030675
49	7	7.0034939
64	8	7.9974201
81	9	9.0019381



Classifying Diabetes with ANNs

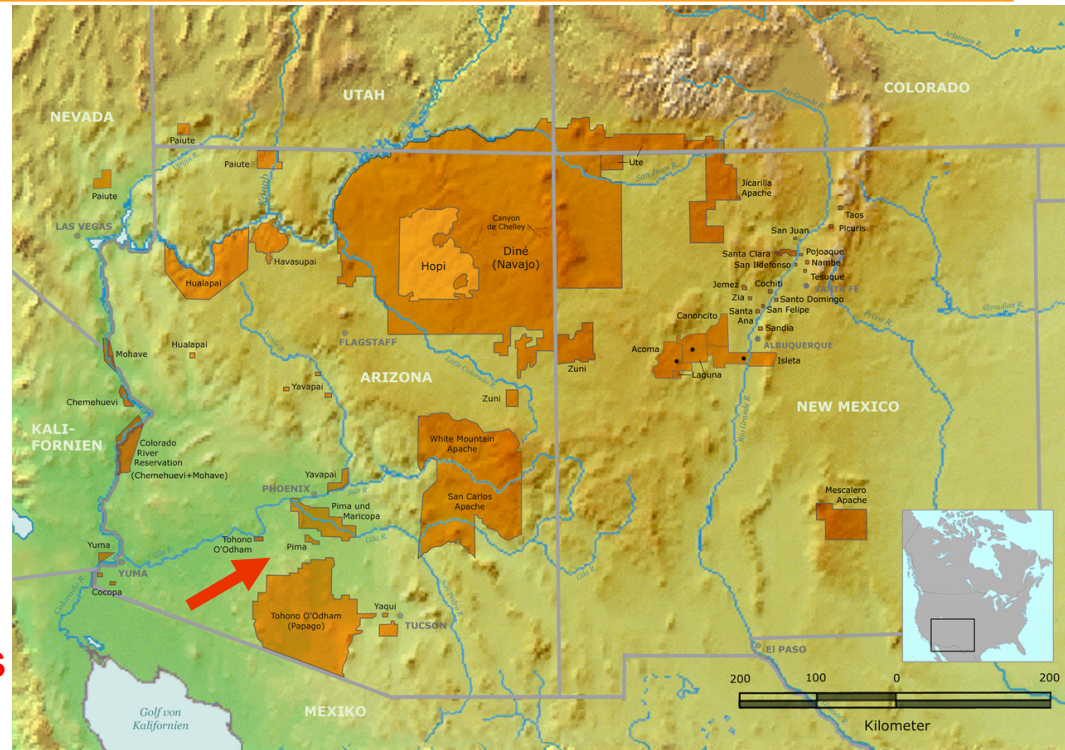


Diabetes classification – Pima Indians

Data from 768 women

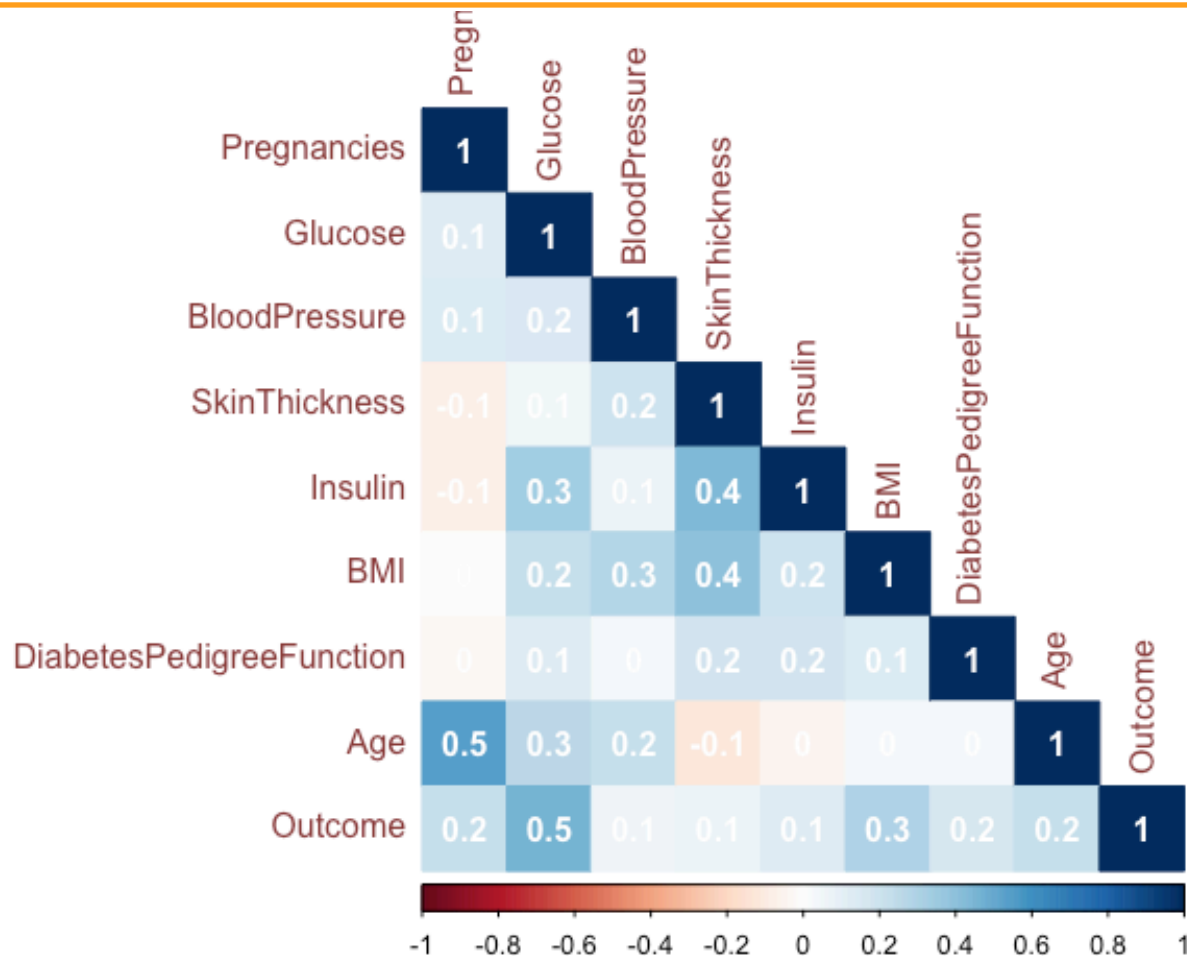
Input
variables

- Pregnancy
- Glucose
- Diastolic blood pressure
- Skin thickness
- Insulin
- BMI
- Diabetes pedigree function
- Age
- Outcome variable – Diabetes



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	Outcomec
1	6	148	72	35	0	33.6	0.627	50	1	diabetes
2	1	85	66	29	0	26.6	0.351	31	0	no diabetes
3	8	183	64	0	0	23.3	0.672	32	1	diabetes
4	1	89	66	23	94	28.1	0.167	21	0	no diabetes
5	0	137	40	35	168	43.1	2.288	33	1	diabetes
6	5	116	74	0	0	25.6	0.201	30	0	no diabetes

Diabetes classification – correlation matrix



```
M <- cor(diabetes.df[,1:9])
corrplot(M, method="color", tl.col = "indianred4",
          number.digits = 1, addCoef.col = "white", type="lower")
```

Normalize data in range 0 to 1

```
normalize <- function(x) {  
  return ((x - min(x)) / (max(x) - min(x)))  
}  
diabetes.df_norm <- as.data.frame(lapply(diabetes.df, normalize))
```

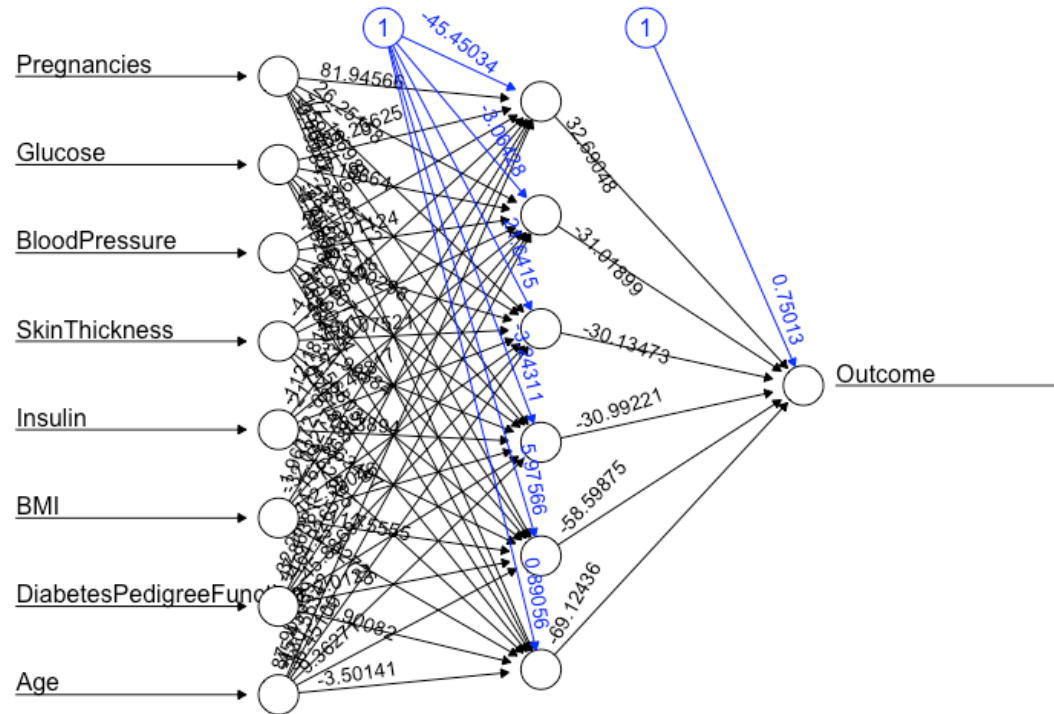
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	0.35294118	0.7437186	0.5901639	0.3535354	0.00000000	0.5007452	0.234415030	0.48333333	1
2	0.05882353	0.4271357	0.5409836	0.2929293	0.00000000	0.3964232	0.116567037	0.16666667	0
3	0.47058824	0.9195980	0.5245902	0.00000000	0.00000000	0.3472429	0.253629377	0.18333333	1
4	0.05882353	0.4472362	0.5409836	0.2323232	0.11111111	0.4187779	0.038001708	0.00000000	0
5	0.00000000	0.6884422	0.3278689	0.3535354	0.19858156	0.6423249	0.943637916	0.20000000	1
6	0.29411765	0.5829146	0.6065574	0.00000000	0.00000000	0.3815201	0.052519214	0.15000000	0

Training / Test data

```
index <- sample(1:nrow(diabetes.df_norm),round(0.90*nrow(diabetes.df_norm)))  
trainset <- diabetes.df_norm[index,]      # 90% training data  
testset <- diabetes.df_norm[-index,]     # 10% test data
```

ANN within R neuralnet

```
nn <- neuralnet(Outcome ~
Pregnancies+Glucose+BloodPressure+SkinThickness+Insulin+BMI+Diabetes
PedigreeFunction+Age,
data=trainset, hidden=c(6), linear.output=FALSE, threshold=0.01,
lifesign='full')
```



Error: 37.668814 Steps: 44984

© 2019 All rights reserved | **Confidential** | For Syneos Health® use only

Confusion Matrix

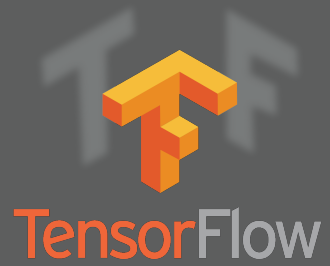
		Prediction	
		No Diabetes	Diabetes
Actual	No Diabetes	46	9
	Diabetes	5	17

	actual	prediction
1	1	0
2	1	1
3	0	0
4	1	1
5	0	0
6	0	0
7	0	0
8	0	0
9	1	1
10	0	0
11	1	0

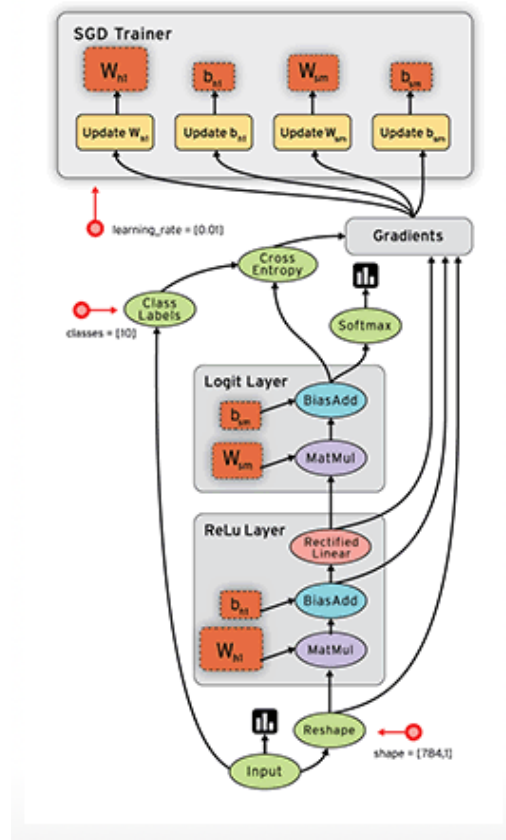
```
nn.results <- compute(nn, testset)
results     <- data.frame(actual = testset$Outcome,
                           prediction = nn.results$net.result)
```

$$\text{False/Positive rate} = \frac{46+17}{46+17+5+9} = 0.82$$

$$\text{MSE} = 0.17$$



TensorFlow with R



TensorFlow



- TensorFlow displays mathematical operations in the form of a graph
- The graph represents the sequential flow of all operations to be performed by TensorFlow
- Speech recognition
- Google Photos
- Developed by Google, now public domain

→ **“A general purpose numerical computing library”**

Classification of breast cancer with TensorFlow

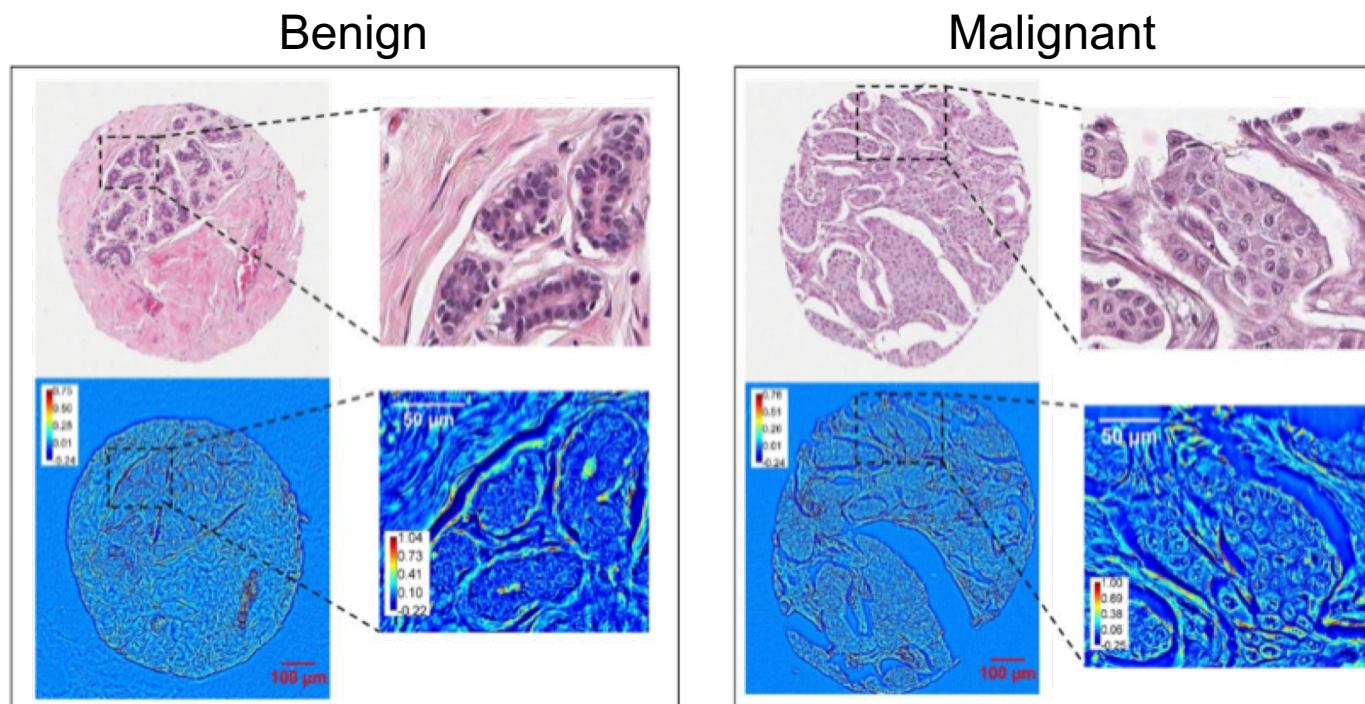
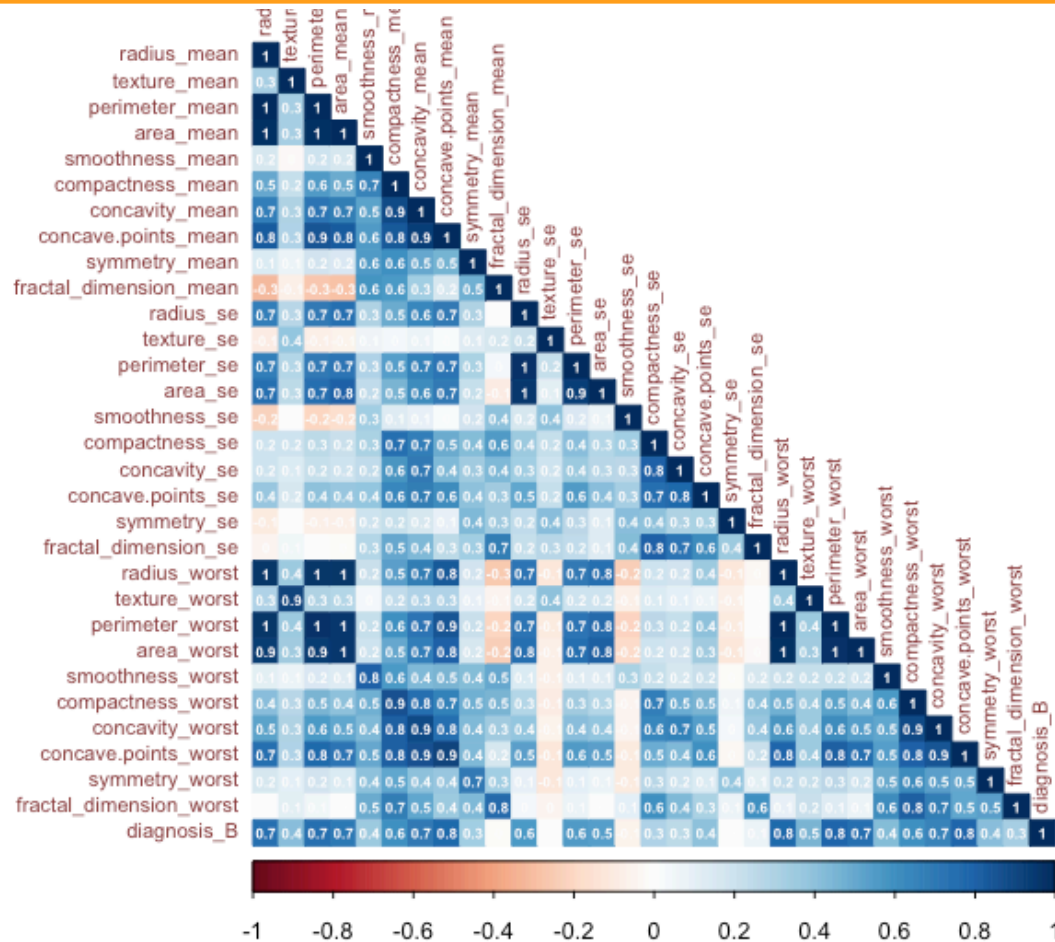


Image: Phys — SPIE: International Society for Optics and Photonics

- Dataset with 569 records and 31 features

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
1	M	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.300100
2	M	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086900
3	M	19.690	21.25	130.00	1203.0	0.10960	0.15990	0.197400

A look at the data



```
M <- cor(df)
```

```
corrplot(M, method="color", number.digits = 1, addCoef.col = "white", number.cex= 0.5, tl.cex = 0.7, type = "lower")
```

R package *Keras*

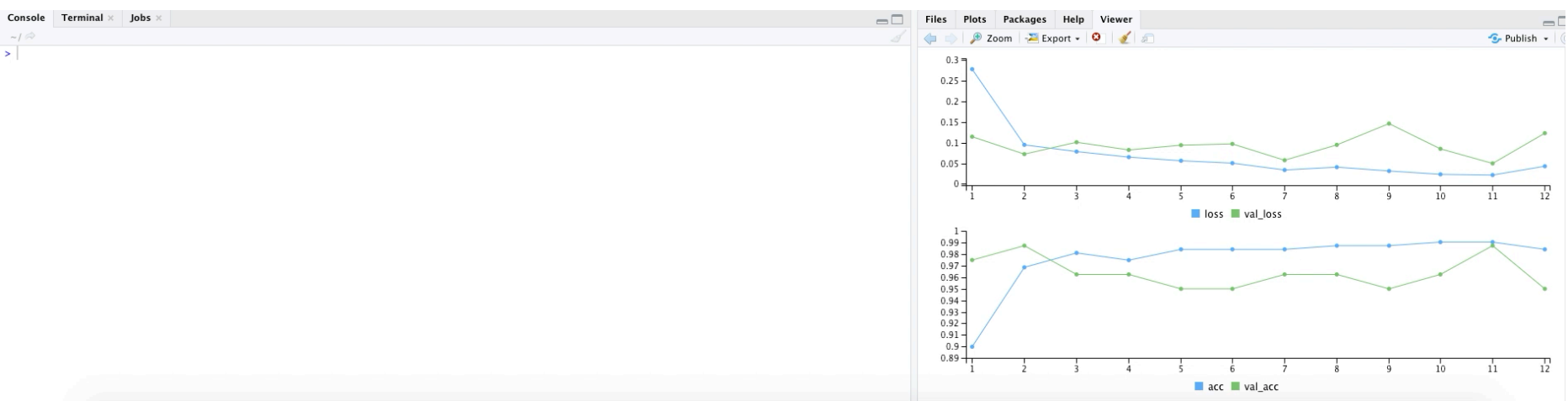
```
model <- keras_model_sequential()
model
%>% layer_dense(units = 256, activation="relu", input_shape = ncol(X_train))
%>% layer_dropout(rate = 0.4)
%>% layer_dense(units = 75, activation = "relu")
%>% layer_dropout(rate = 0.3)
%>% layer_dense(units = 2, activation = "sigmoid")

model %>% compile( optimizer = 'adam',
                  loss =sparse_categorical_crossentropy',
                  metrics = c('accuracy'))

model %>% fit(X_train,y_train,epochs=12,batch_size=5,validation_split=0.2)
```

Training the ANN with TensorFlow

```
model %>%  
fit(X_train,y_train,epochs=12,batch_size=5,validation_split=0.2)
```



Prediction with TensorFlow

```
predictions <- model %>% predict_classes(X_test)
df.pred <- data.frame(predictions)
```

Confusion Matrix

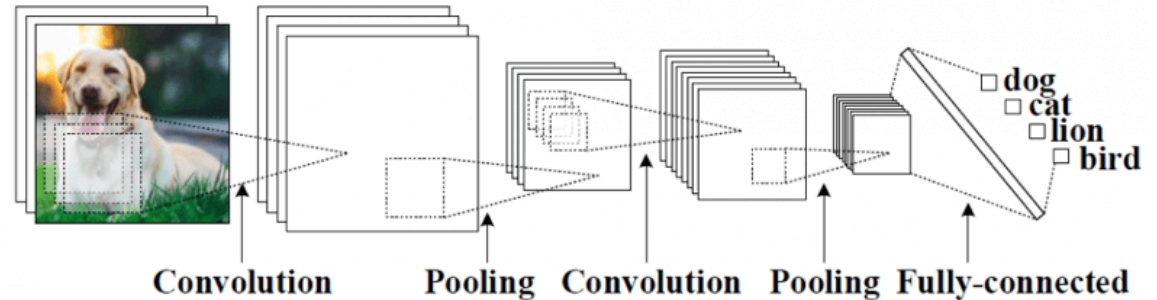
		Prediction	
		benign	malignant
Actual	benign	103	2
	malignant	3	62

False/positive rate = 97%

Convolutional Neural Networks

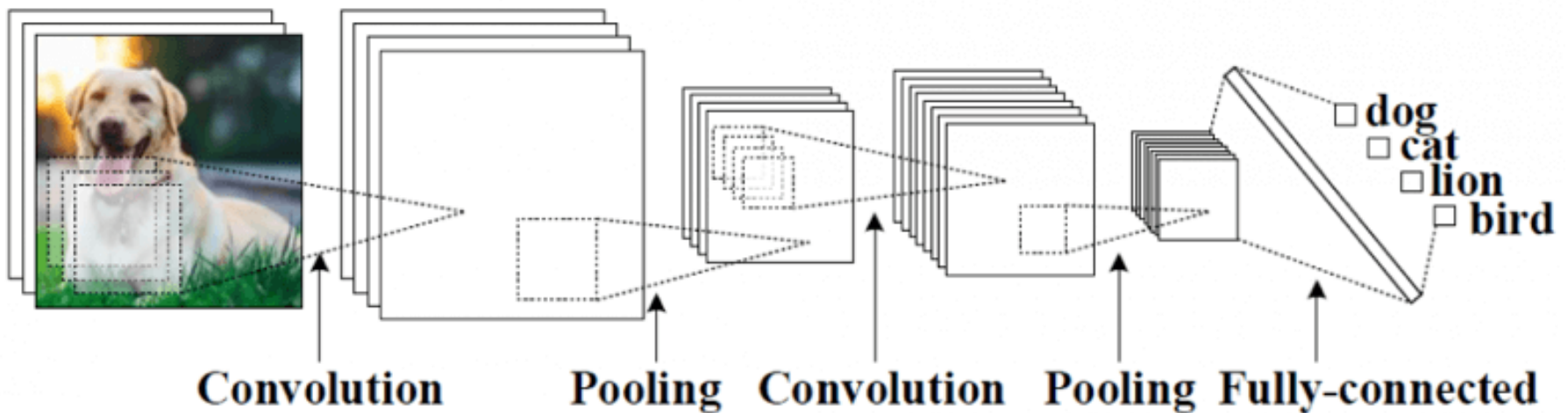
Object detection and identification

Speech and natural language processing



Convolutional Neural Networks (CNN)

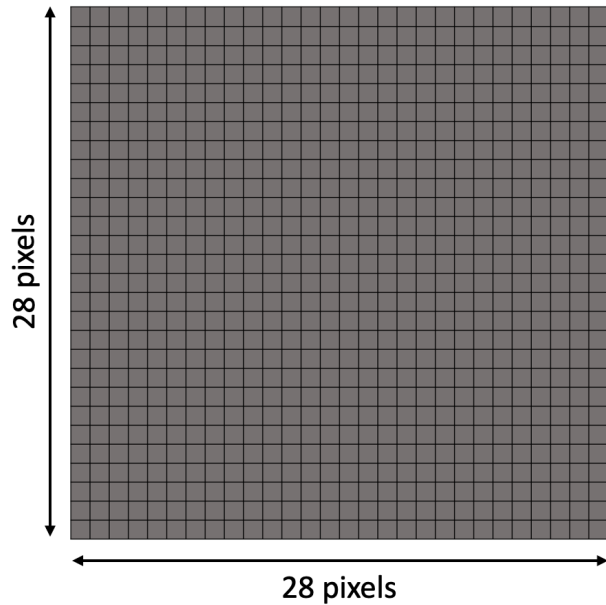
- Maintain spatial integrity of input images → 2d or 3d matrix as input
- Feature extraction through convolutional filters
- Dimensionality reduction via pooling
- Fully connected ANN for categorization



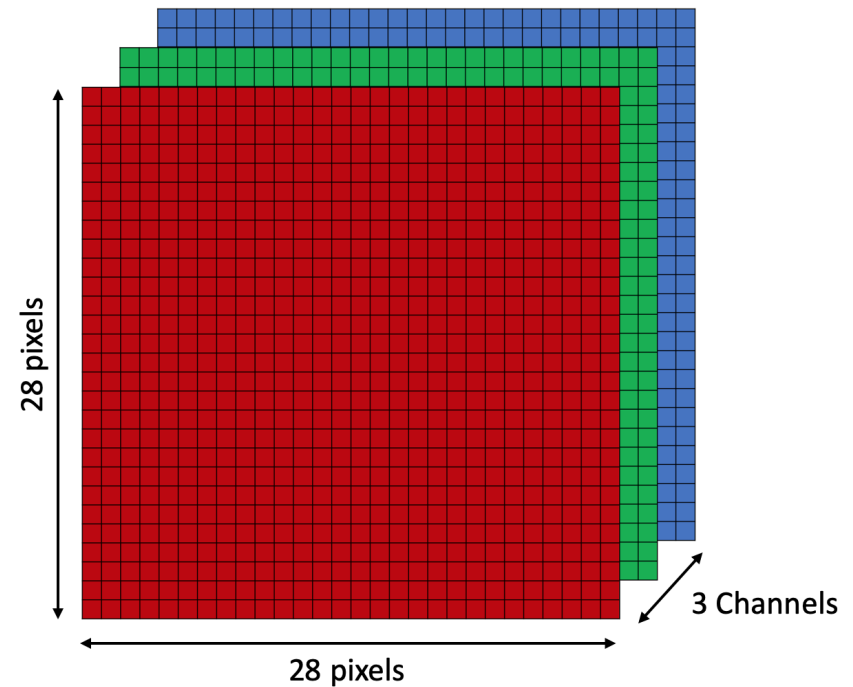
Maintaining of spatial integrity

- Conventional ANN would converted to a 1D list of pixel values
- Loss of spatial integrity (e.g. shadows or edges)

Greyscale (28 x 28 x 1)



RGB/Colour (28 x 28 x 3)



Convolutional filter

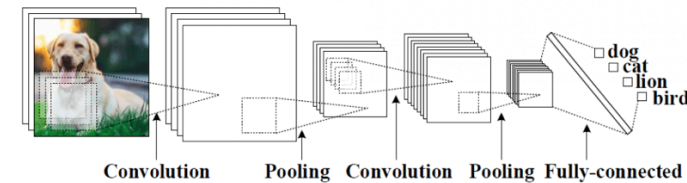
Original Image

1	1	1	1	1	1
1	1	1	1	1	1
0	0	1	1	0	0
0	0	1	1	0	0
0	0	1	1	0	0
0	0	1	1	0	0

Feature Map

x1	x1	x1
x1	x1	x1
x0	x0	x0

Convolutional filter with weights



Pooling

Feature Map

6	6	6	6
4	5	5	4
2	4	4	2
2	4	4	2

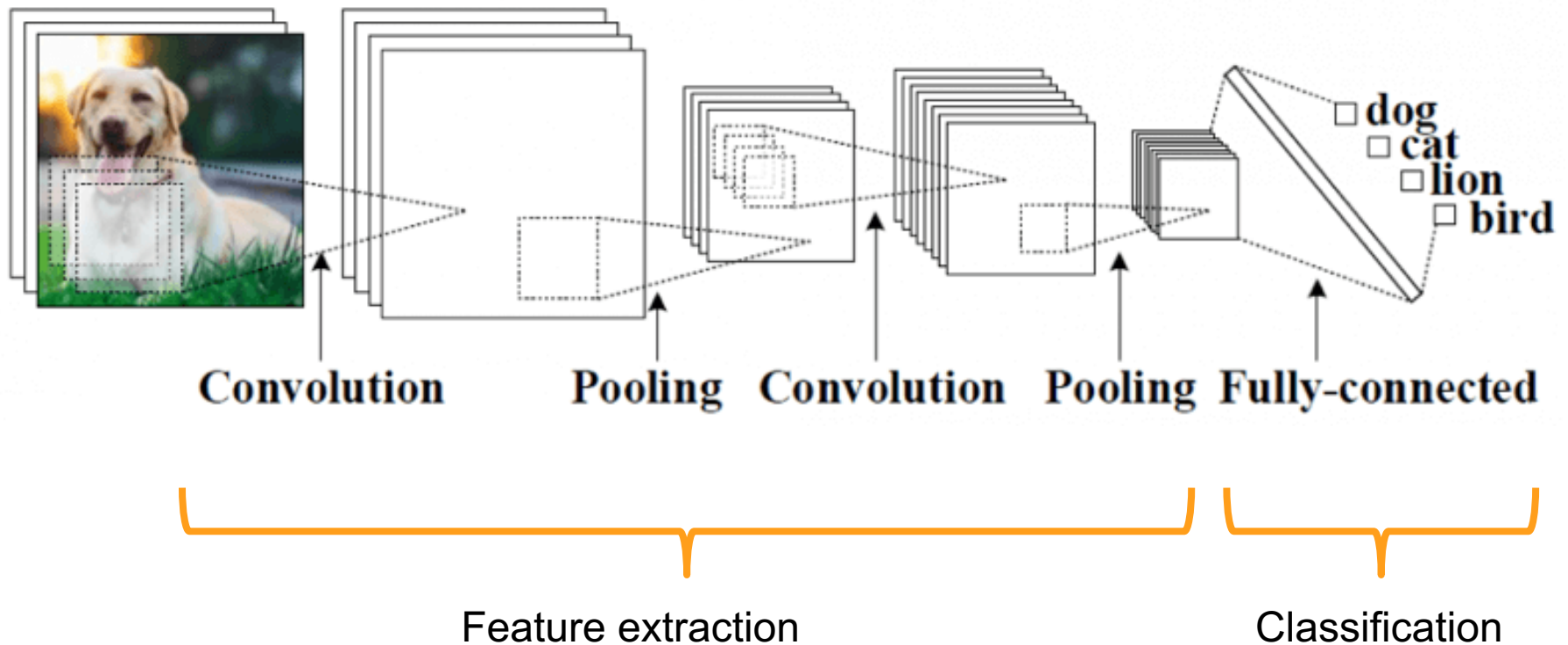
Max
Pooling

Average
Pooling

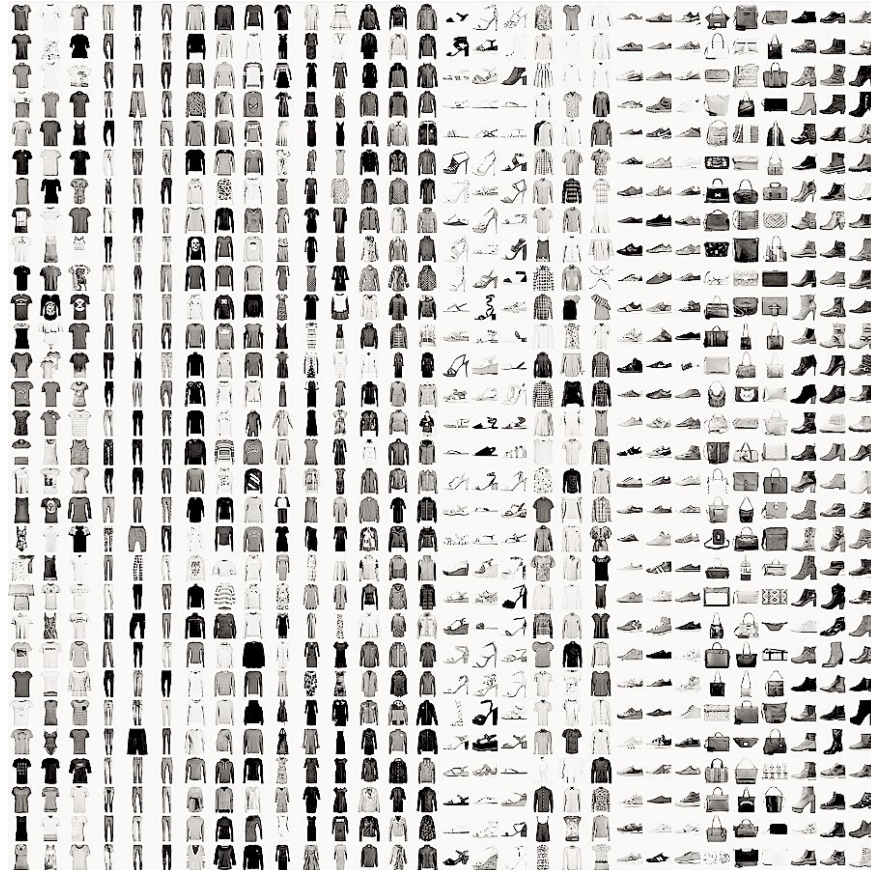
Sum
Pooling

Pooling reduces the feature map size without loss of information

Convolutional Neural Networks (CNN)



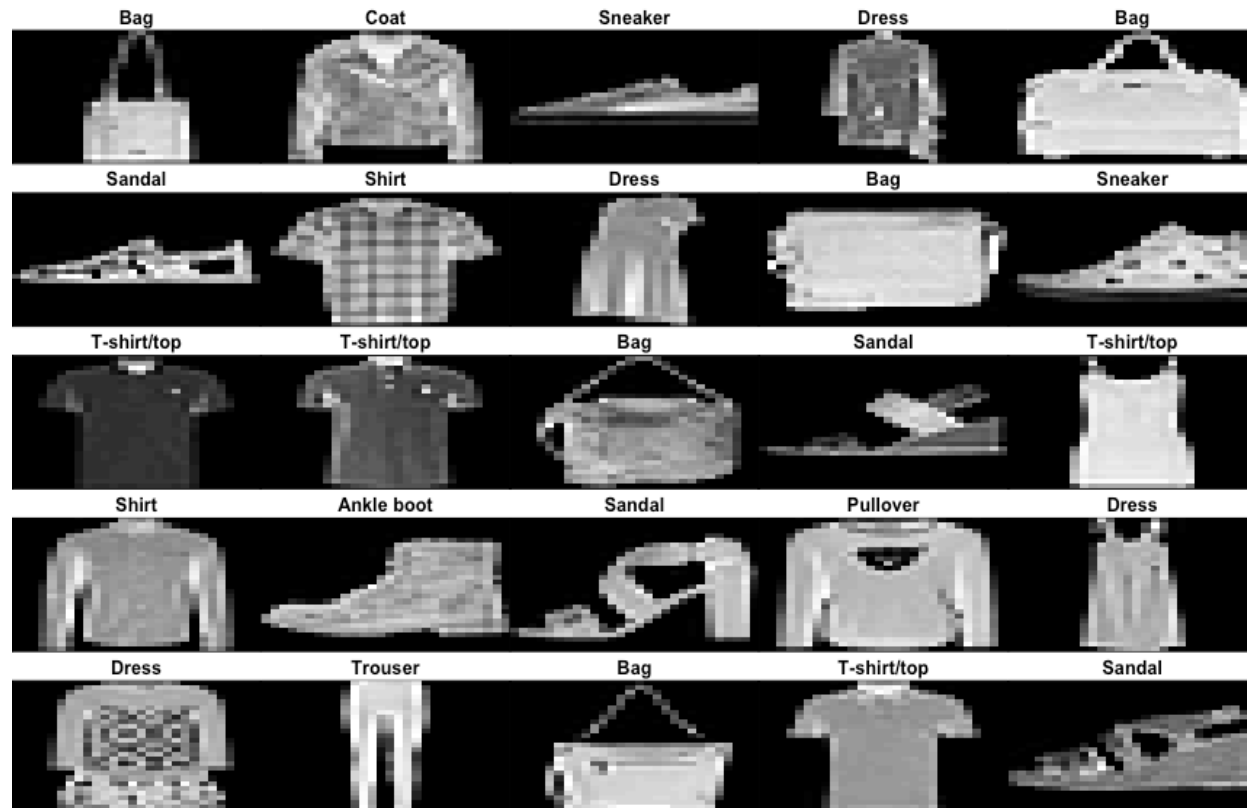
„Hello machine learning world“



Fashion MNIST¹ dataset – Object detection

70,000 grayscale images (28 × 28 pixels) in 10 categories.

T-shirt/top	0
Trouser	1
Pullover	2
Dress	3
Coat	4
Sandal	5
Shirt	6
Sneaker	7
Bag	8
Ankle boot	9



¹MNIST - Modified National Institute of Standards and Technology database

Prepare Fashion MNIST dataset

```
library(keras)
```

Get dataset

```
fashion_mnist <- dataset_fashion_mnist()
```

Training / Test data

```
c(train_images, train_labels) %<-% fashion_mnist$train  
c(test_images, test_labels) %<-% fashion_mnist$test
```

scale pixel values to range of 0 to 1

```
train_images <- train_images / 255  
test_images <- test_images / 255
```



Building the CNN

#Build model

```
model <- keras_model_sequential()
model %>%
  layer_reshape(c(28,28,1)) %>%
  layer_conv_2d(input_shape=c(28,28,1), filter = 32,
    kernel_size = c(3,3), activation='relu') %>%
  layer_max_pooling_2d(pool_size=c(2,2)) %>%
  layer_flatten() %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax')
```

#Compile Model

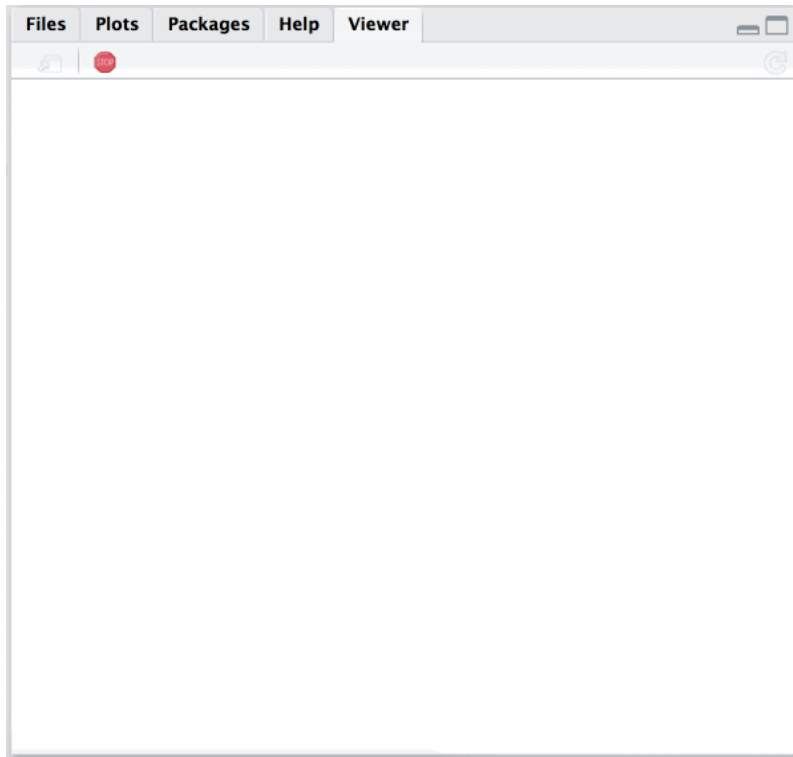
```
model %>% compile(optimizer = 'adam',
  loss = 'sparse_categorical_crossentropy',
  metrics = c('accuracy'))
```

#Start Training process

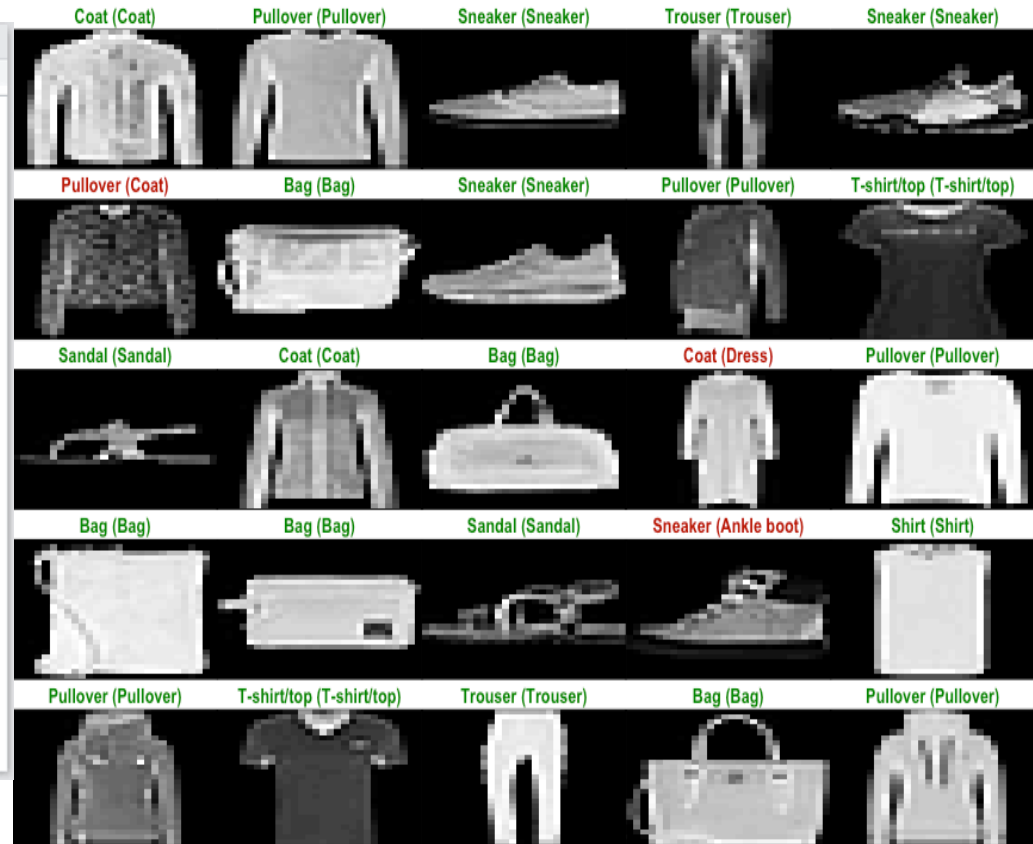
```
model %>% fit(train_images, train_labels, epochs = 8)
```

Training process in R

```
model %>% fit(train_images, train_labels, epochs = 8)
```

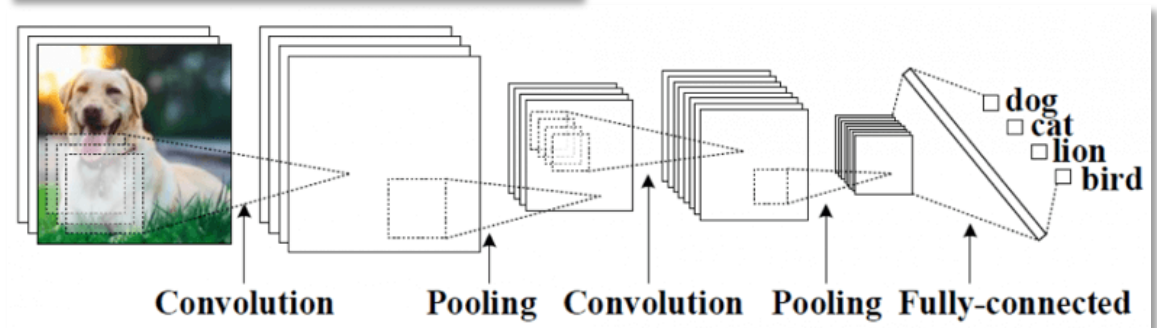
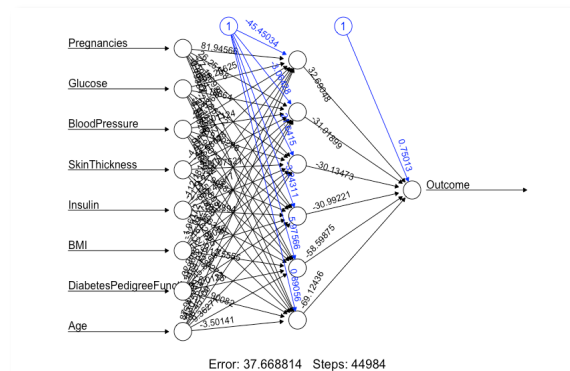


Accuracy of >90%



Summary

- Introduction into ANNs
- Artificial Neural Networks with R's `neuralnet`
- Diabetes classification with simple ANNs
- Breast cancer classification with TensorFlow
- Convolutional Neural Networks with R's `keras`



Thank you for your attention!

Questions to:

Thomas Wollseifen

Syneos Health Germany GmbH

Email: thomas.wollseifen@syneoshealth.com

Shortening the Distance From Lab to Life[®].
