Brain Haemorrhage Diagnosis - Using Lenet based Deep Learning Model

```
(*change the path to the data source in your pc*)
```

In[1]:=
```
infected = FileNames["*.png",
   "E:\\COURSES\\Wolfram\\Brain Tumor Images Dataset\\training_set\\hemmorhage_data"];
uninfected = FileNames["*.png",
   "E:\\COURSES\\Wolfram\\Brain Tumor Images Dataset\\training_set\\non_hemmorhage
     _data"];
```

In[3]:=
```
infectedIMG = File /@ infected;
uninfectedIMG = File /@ uninfected;
```

In[5]:=
```
Length[infectedIMG]
```

Out[5]= 70

In[6]:=
```
infectedvalues = Table[True, Length[infected]];
Length[uninfected]
```

Out[7]= 70

In[8]:=
```
uninfectedvalues = Table[False, Length[uninfected]];
```

In[29]:=
```
data = RandomSample[AssociationThread[infectedIMG -> infectedvalues]];
traininglength = Length[data] * .75
```

Out[30]= 52.5

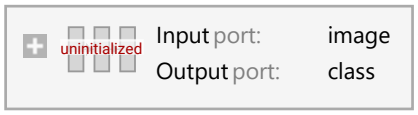
In[11]:=
```
trainingdata = data[[1 ;; 52]];
validationdata = data[[53 ;;]];
```

In[13]:=
```
dims = {135, 135}
```

Out[13]= {135, 135}

In[14]:=
```
lenet = NetChain[{ResizeLayer[dims],
    ConvolutionLayer[20, 5], Ramp, (*Takes out the the not useful features*)
    PoolingLayer[2, 2], (*Downsamples*)ConvolutionLayer[50, 5], Ramp,
    (*Takes out the the not useful features*)PoolingLayer[2, 2], (*Downsamples*)
    FlattenLayer[], 500, (*Makes features into feature vector"*)Ramp, 2,
    (*Takes out the the not useful features-True or false*)SoftmaxLayer[]},
   (*Turns the vector into probabilities*)
   "Output" → NetDecoder[{"Class", {True, False}}], (*Tensor into true or false*)
   "Input" → NetEncoder["Image"](*Turns image into numbers*)]
```

Out[14]=

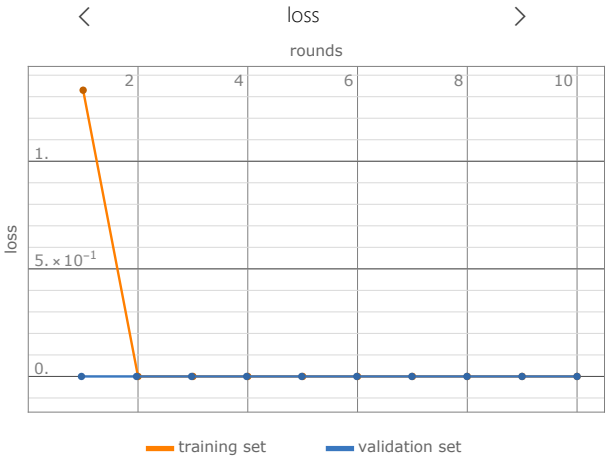


In[15]:=
```
results =
 NetTrain[lenet, Normal[trainingdata], All,
  ValidationSet -> Normal[validationdata], MaxTrainingRounds -> 10,
  TargetDevice -> "CPU"]
```

Out[15]=

NetTrain Results

summary	batches: 10, rounds: 10, time: 25s, examples/s: 21.
data	training examples: 52, validation examples: 18, processed examples: 520, skipped examples: 0
method	ADAM optimizer, batch size 52, CPU
round	loss: 0., error: 0%
validation	loss: 0., error: 0%



In[16]:=
```
augment = ImageAugmentationLayer[{135, 135},
  "Input" → NetEncoder[{"Image", {139, 139}}], "Output" → NetDecoder["Image"]]
```

Out[16]=

ImageAugmentationLayer

Input:	image
Output:	image

In[17]:=
```
dims2 = {139, 139}

lenet2 = NetChain[{ResizeLayer[dims2], ImageAugmentationLayer[{135, 135}],
   ConvolutionLayer[20, 5], Ramp, PoolingLayer[2, 2], ConvolutionLayer[50, 5],
   Ramp, PoolingLayer[2, 2], FlattenLayer[], 500, Ramp, 2, SoftmaxLayer[]},
  "Output" → NetDecoder[{"Class", {True, False}}], "Input" → NetEncoder["Image"]]
```

Out[17]= {139, 139}

Out[18]=

NetChain

uninitialized

Input port:	image
Output port:	class

In[19]:=
```
results2 =
 NetTrain[lenet2, Normal[trainingdata], All,
  ValidationSet -> Normal[validationdata], MaxTrainingRounds -> 7]
```

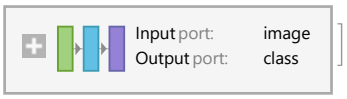
Out[19]= NetTrainResultsObject

NetTrain Results

summary	batches: 91, rounds: 7, time: 31s, examples/s: 12.
data	training examples: 52, validation examples: 18, processed examples: 364, skipped examples: 0
method	ADAM optimizer, batch size 4, CPU
round	loss: 0., error: 0%
validation	loss: 0., error: 0%

In[20]:=
```
trained = results2["TrainedNet"]
```

Out[20]= NetChain

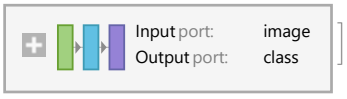


In[21]:=
```
Export["augmentnet.wlnet", trained]
```

Out[21]= augmentnet.wlnet

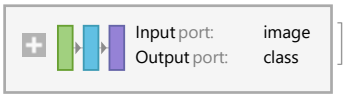
In[22]:=
```
trained2 = results["TrainedNet"]
```

Out[22]= NetChain



In[23]:=
```
trained = Import["E:\\COURSES\\Wolfram\\Projects\\augmentnet.wlnet"]
```

Out[23]= NetChain



In[24]:=
```
WLNetImport[E : \\ COURSES \\ Wolfram \\ Projects \\ augmentnet.wlnet, Net]
```

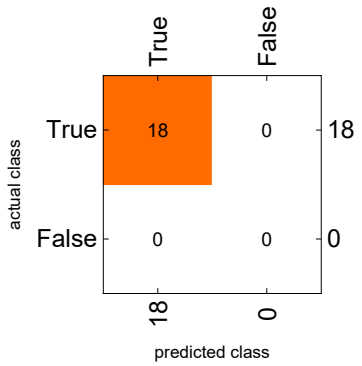
In[24]:=
```
testset = Import /@ Keys[validationdata];
```

In[25]:=
```
visual = ClassifierMeasurements[trained,
  Normal[RandomSample[AssociationThread[testset → Values[validationdata]]]]]
```

Out[25]=

Classifier Measurements

Classifier method	Net
Number of test examples	18
Accuracy	(100.0000000000000(00 ± 22))%
Accuracy baseline	(100.0000000000000(00 ± 22))%
Single evaluation time	31.1 ms/example
Batch evaluation speed	11.4 examples/s

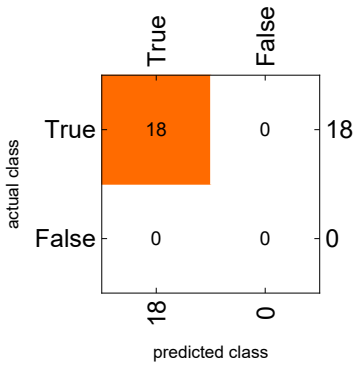


In[26]:=
```
visual /@ {"Accuracy", "FScore", "ConfusionMatrixPlot", "Precision", "Recall",
   "Sensitivity", "FalsePositiveRate", "WorstClassifiedExamples"} // TableForm
```

Out[26]//TableForm=

1.

⟨|True → 1., False → Indeterminate|⟩

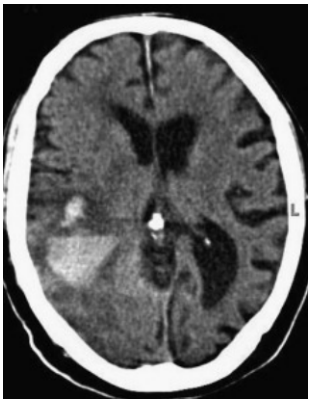


⟨|True → 1., False → Indeterminate|⟩

⟨|True → 1., False → Indeterminate|⟩

⟨|True → 1., False → Indeterminate|⟩

⟨|True → Indeterminate, False → 0.|⟩



→ True

→ True