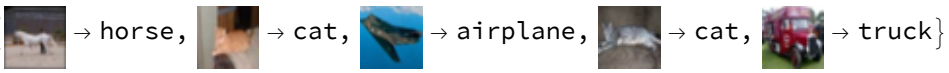
CIFAR-10

In[7]:=
```
cifar10 = ResourceObject["CIFAR-10"];
trainingData = ResourceData[cifar10, "TrainingData"];
testData = ResourceData[cifar10, "TestData"];
RandomSample[trainingData, 5]
```

Out[8]=  → horse, → cat, → airplane, → cat, → truck

In[9]:=
```
classes = Union@Values[trainingData]
```

Out[9]= {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck}

Logistic Regression

In[46]:=
```
training1000 = RandomSample[trainingData, 1000];
testing100 = RandomSample[testData, 100]
```

Out[47]=



```
AbsoluteTiming[(logit = Classify[training1000, Method → "LogisticRegression"])]
```

In[53]:=
```
predLabel = logit[Keys@testing100];
```

In[54]:=
```
realLabel = Values[testing100];
```

In[55]:=
```
Counts[MapThread[Equal, {predLabel, realLabel}]]
```

Out[55]=
```
⟨|True → 78, False → 22|⟩
```

In[57]:=
```
accuracy = 78/100 // N
```

Out[57]=
```
0.78
```

In[73]:=
```
logitFunction = Association @@ Normal[logit]
```

In[74]:=
```
Keys[logitFunction]
```

Out[74]=
```
{Basic, Input, Output, Combiner, Decision, Models, Log}
```

In[80]:=
```
ts = logitFunction["Models"][[1]]["Theta"];
```

In[81]:=
```
Dimensions[ts]
```

Out[81]=
```
{10, 440}
```

In[67]:=
```
ClassifierInformation[logit]
```

Out[67]=

Classifier information

Method	Logistic regression
Number of classes	10
Number of features	1
Number of training examples	1000
L1 regularization coefficient	0
L2 regularization coefficient	10.

In[66]:=
```
ClassifierInformation[logit, "Models"]
```

Out[66]=
```
Missing[PropertyNotAvailable, Models]
```

Naive CNN model

In[23]:=
```
naiveCNN = NetChain[{ConvolutionLayer[20, 5], 10, SoftmaxLayer[]},
  "Output" → NetDecoder[{"Class", classes}],
  "Input" → NetEncoder[{"Image", {32, 32}}]]
```

Out[23]=

NetChain

	Input	image 3–tensor (size: 3 × 32 × 32)
1	ConvolutionLayer	3–tensor (size: 20 × 28 × 28)
2	LinearLayer	vector (size: 10)
3	SoftmaxLayer	vector (size: 10)
	Output	class

(uninitialized)

In[24]:=
```
cnn1layer =
  NetTrain[naiveCNN, trainingData, ValidationSet → testData, MaxTrainingRounds → 3];
```

In[25]:=
```
predClass = cnn1layer /@ (Keys@testData)
```

Out[25]=
```
{frog, airplane, ship, ship, ship, ship, ship, ship, airplane, airplane,
 ship, ship, truck, airplane, ship, airplane, ship, ship, ship, ship, dog,
 ship, ship, ship, ··· 9952 ···, automobile, truck, truck, truck, ship,
 frog, truck, ship, truck, truck, truck, ship, truck, truck, horse, truck,
 automobile, frog, truck, automobile, truck, automobile, automobile, truck}
```

large output | show less | show more | show all | set size limit...

In[26]:=
```
realClass = Values[testData]
```

Out[26]=
```
{airplane, airplane, airplane, airplane, airplane, airplane, airplane, airplane,
 airplane, airplane, airplane, airplane, airplane, airplane, airplane, airplane,
 airplane, airplane, airplane, airplane, airplane, airplane, ··· 9958 ···,
 truck, truck, truck, truck, truck, truck, truck, truck, truck, truck, truck,
 truck, truck, truck, truck, truck, truck, truck, truck, truck, truck, truck}
```

large output | show less | show more | show all | set size limit...

In[27]:=
```
Counts[MapThread[Equal, {predClass, realClass}]]
```

Out[27]=
```
⟨| False → 6100, True → 3900 |⟩
```

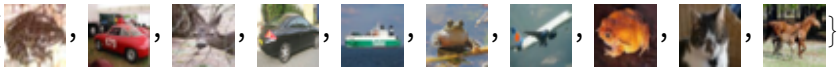
In[28]:=
```
accuracy = 3900 / 10 000 // N
```

Out[28]=
```
0.39
```

In[29]:=
```
randomTestData = RandomSample[testData, 10]
```

Out[29]=


In[30]:=
```
Keys@randomTestData
```

Out[30]=


In[31]:=
```
cnn1layer[Keys@randomTestData]
```

Out[31]=
```
{cat, truck, dog, automobile, ship, deer, ship, cat, cat, frog}
```

LeNet model

In[32]:=
```
lenet = NetChain[{ConvolutionLayer[20, 5], Ramp, PoolingLayer[2, 2],
   ConvolutionLayer[50, 5], Ramp, PoolingLayer[2, 2], FlattenLayer[], 500,
   Ramp, 10, SoftmaxLayer[]}, "Output" → NetDecoder[{"Class", classes}],
  "Input" → NetEncoder[{"Image", {32, 32}}]]
```

Out[32]=

NetChain

		image
	Input	3–tensor (size: 3 × 32 × 32)
1	ConvolutionLayer	3–tensor (size: 20 × 28 × 28)
2	Ramp	3–tensor (size: 20 × 28 × 28)
3	PoolingLayer	3–tensor (size: 20 × 14 × 14)
4	ConvolutionLayer	3–tensor (size: 50 × 10 × 10)
5	Ramp	3–tensor (size: 50 × 10 × 10)
6	PoolingLayer	3–tensor (size: 50 × 5 × 5)
7	FlattenLayer	vector (size: 1250)
8	LinearLayer	vector (size: 500)
9	Ramp	vector (size: 500)
10	LinearLayer	vector (size: 10)
11	SoftmaxLayer	vector (size: 10)
	Output	class

(uninitialized)

In[34]:=
```
lenet = NetTrain[lenet, trainingData, ValidationSet → testData, MaxTrainingRounds → 3]
```

Out[34]=

NetChain

		image
	Input	3–tensor (size: 3 × 32 × 32)
1	ConvolutionLayer	3–tensor (size: 20 × 28 × 28)
2	Ramp	3–tensor (size: 20 × 28 × 28)
3	PoolingLayer	3–tensor (size: 20 × 14 × 14)
4	ConvolutionLayer	3–tensor (size: 50 × 10 × 10)
5	Ramp	3–tensor (size: 50 × 10 × 10)
6	PoolingLayer	3–tensor (size: 50 × 5 × 5)
7	FlattenLayer	vector (size: 1250)
8	LinearLayer	vector (size: 500)
9	Ramp	vector (size: 500)
10	LinearLayer	vector (size: 10)
11	SoftmaxLayer	vector (size: 10)
	Output	class

In[35]:=
```
predClass = lenet /@ (Keys@testData)
```

Out[35]=
```
{ship, airplane, ship, airplane, airplane, airplane, airplane, airplane, airplane,
 airplane, airplane, dog, airplane, airplane, ship, airplane, airplane,
 ··· 9966 ···, truck, automobile, truck, truck, truck, truck, truck, horse,
 automobile, truck, truck, truck, automobile, truck, truck, truck, truck}
```

large output | show less | show more | show all | set size limit...

In[36]:=
```
realClass = Values[testData]
```

Out[36]=
```
{airplane, airplane, airplane, airplane, airplane, airplane, airplane, airplane,
 airplane, airplane, airplane, airplane, airplane, airplane, airplane,
 airplane, airplane, ··· 9966 ···, truck, truck, truck, truck, truck, truck,
 truck, truck, truck, truck, truck, truck, truck, truck, truck, truck, truck}
```

large output | show less | show more | show all | set size limit...

In[37]:=
```
Counts[MapThread[Equal, {predClass, realClass}]]
```

Out[37]=
```
⟨| False → 3883, True → 6117 |⟩
```

In[38]:=
```
accuracy = 6117 / 10 000 // N
```

Out[38]=
```
0.6117
```