

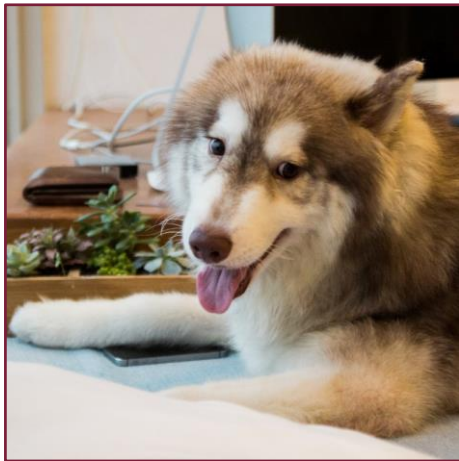
Convolutional Neural Network

Systems Programming
Department of Computer Science and Engineering
Sogang University

Image Classification

1

Image classification



```

text", r.resetText||n.data("resetText", n[i]()), n[i](r[e]||this.op
s(t).attr(t, t):n.removeClass(t).removeAttr(t)}, 0)}, t.prototype.t
cons-radio"'); e&&e.find(".active").removeClass("active"), this.$
function(n){return this.each(function(){var r=e(this), i=r.data("b
,s)), n=="toggle"?i.toggle():n&&i.setState(n)}), e.fn.button.defa
button.noConflict=function(){return e.fn.button=n, this}, e(docume
tion(t){var n=e(t.target); n.hasClass("btn")||(n=n.closest(".btn
var t=function(t, n){this.$element=e(t), this.$indicators=this.$el
se=="hover"&&this.$element.on("mouseenter", e.proxy(this.pause, th
function(t){return t||(this.paused=!1), this.interval&&clearInter
this.interval=setInterval(e.proxy(this.next, this), this.options.in
element.find(".item.active"), this.$items=this.$active.parent().c
or n=this.getActiveIndex(), r=this; if(t>this.$items.length-1||t<0
n(){r.to(t)}:n=t?this.pause().cycle():this.slide(t>n?"next":"
(0), this.$element.find(".next, .prev").length&&e.support.transit
s.cycle(!0)), clearInterval(this.interval), this.interval=null, th
ext")), prev:function(){if(this.sliding)return; return this.slide(
.active"), i=n||r[t](), s=this.interval, o=t=="next"?left:"right"
se(), i=i.length?i:this.$element.find("item")[u](), f=e.Event("sl
rn; this.$indicators.length&&(this.$indicators.find("active").r
a.$indicators.removeClass("active"), a.addClass("act
){this.$element.trigger(e.Event("transitionend"))return; i.ad
t.one(e.support.transitionend, function(){i.removeClass([t, o].jo
pin(" ")), a.sliding=!1, setTimeout(function(){a.$element.trigger(
)}return; r.removeClass("active"), i.addClass("active"), this.slid
i}); var n=e.fn.carousel; e.fn.carousel=function(n){return this.ea
e.fn.carousel.defaults, typeof n=="object"&&n, o=typeof n=="strin
"?i.to(n):o?i[o]():s.interval&&i.pause().cycle()}), e.fn.carous
l.Constructor=t, e.fn.carousel.noConflict=function(){return e.fn.
slide], [data-slide-to], function(t){var n=e(this), r=i.e(n.attr(
')), s=e.extend({}, i.data(), n.data()), o=i.carousel(s), (o=n.attr("
eventDefault()))}(window.jQuery), !function(e){"use strict"; var t
collapse.defaults, n, this.options.parent&&(this.$parent=e(this.op
structor:t, dimension:function(){var e=this.$element.hasClass("wi
if(this.transitioning||this.$element.hasClass("in"))return; t=th
t&&this.$parent.find(">.accordion-group >.in"); if(r&&r.length)
lapse("hide"), i||r.data("collapse", null)}this.$element[t](0), thi
ons&&this.$element[t](this.$element[0][n]), hide:function(){var t
is.dimension(), this.reset(this.$element[t]()), this.transition("r
function(e){var t=this.dimension(); return this.$element.removeCl
e!=null?"addClass":"removeClass"]("collapse"), this}, transition:
l.transitioning=0, i.$element.trigger(r)}; this.$element.trigger(n

```

Classifier

Cat

Dog

MNIST Classification

Use “04.MNIST.ipynb”

<http://yann.lecun.com/exdb/mnist/>

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.



2

Import modules and tuning parameters

```
[ ] import keras
    from keras.datasets import mnist
    from keras.models import Sequential
    from keras.layers import Dense, Dropout, Flatten, Activation
    from keras.layers import Conv2D, MaxPooling2D
    from keras import backend as K
    from keras.utils.vis_utils import model_to_dot
    from IPython.display import SVG
    %matplotlib inline
    import matplotlib.pyplot as plt
    from sklearn.metrics import confusion_matrix
    import pandas as pd
    import seaborn as sns
```

```
[ ] epochs = 10
    learning_rate=0.01
```

```
[ ] batch_size = 128
    num_classes = 10
```

```
[ ] def plot_images(x, y_true, y_pred=None, size=(5, 5)):
    assert len(x) == len(y_true) == size[0] * size[1]

    fig, axes = plt.subplots(size[0], size[1])
    fig.subplots_adjust(hspace=0.5, wspace=0.1)

    for i, ax in enumerate(axes.flat):
        if x[i].shape[-1] == 1:
            ax.imshow(x[i].reshape(x[i].shape[0], x[i].shape[1]))
        else:
            ax.imshow(x[i])

        if y_pred is None:
            xlabel = "True: {0}".format(y_true[i].argmax())
        else:
            xlabel = "True: {0}, Pred: {1}".format(y_true[i].argmax(),
                                                    y_pred[i].argmax())

        ax.set_xlabel(xlabel)

        ax.set_xticks([])
        ax.set_yticks([])

    plt.show()
```

3

Loading and reshaping the data

```
[ ] (x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
[ ] if len(x_train.shape) < 4:
    x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[2], 1)
    x_test = x_test.reshape(x_test.shape[0], x_train.shape[1], x_train.shape[2], 1)
```

```
[ ] x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')
    x_train /= 255
    x_test /= 255
    print('x_train shape:', x_train.shape)
    print(x_train.shape[0], 'train samples')
    print(x_test.shape[0], 'test samples')
```

```
[ ] y_train = keras.utils.to_categorical(y_train, num_classes)
    y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
[ ] plot_images(x_train[:25], y_train[:25])
```

x_train shape: (60000, 28, 28, 1)

60000 train samples

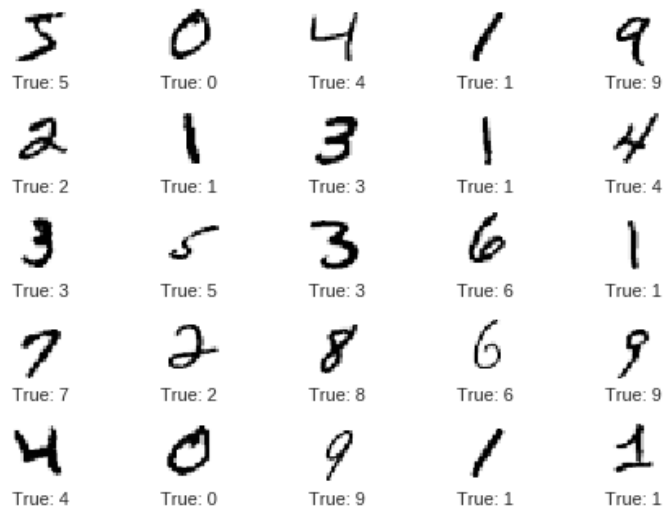
10000 test samples

Download

11493376,

[/mnist.npz](#)

/step

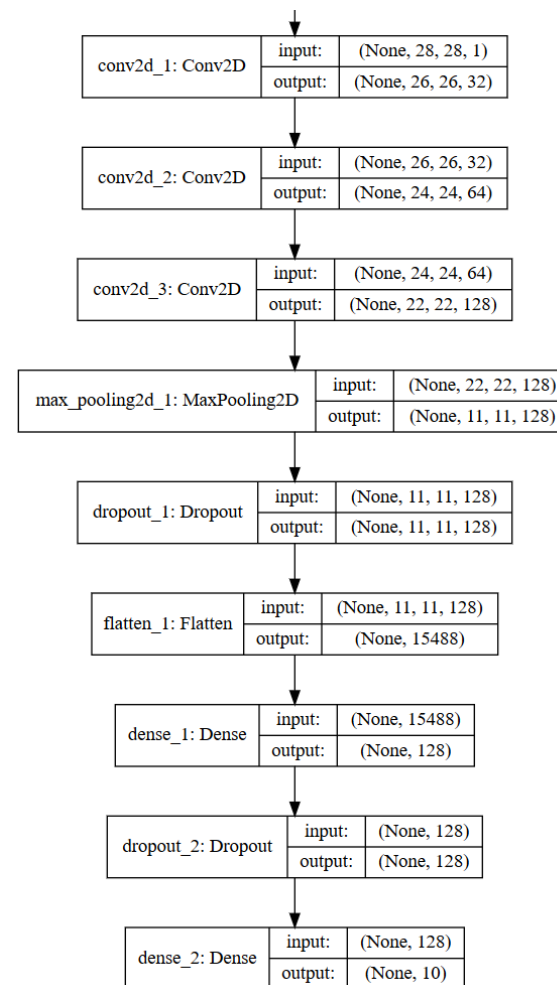


4

Creating the model

```
[ ] model = Sequential()
```

```
[ ] model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
                    input_shape=x_train.shape[1:]))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```



4

Creating the model

```
keras.layers.Conv2D(filter, kernel_size, strides, padding, activation, .....
```

2D convolution layer.

Arguments: filters, kernel_size, strides, padding, activation and so on.

```
model.add(Conv2D(64, (3, 3), activation='relu'))
```

```
keras.layers.MaxPooling2D(pool_size, strides, padding, .....
```

Max pooling operation for spatial data.

Arguments: pool_size, strides, padding and so on.

```
model.add(MaxPooling2D((2, 2))
```

```
keras.layers.Dropout(rate, .....
```

Applies Dropout to the input.

```
keras.layers.Flatten(.....)
```

Flattens the input.

5

Compiling and training the model

```
[ ] optimizer = keras.optimizers.SGD(lr=learning_rate)
```

```
[ ] model.compile(loss=keras.losses.categorical_crossentropy,  
                  optimizer=optimizer,  
                  metrics=['accuracy'])
```

```
[ ] model.fit(x_train, y_train,  
             batch_size=batch_size,  
             epochs=epochs,  
             verbose=1,  
             validation_data=(x_test, y_test))
```

Train on 60000 samples, validate on 10000 samples

```
Epoch 1/10  
60000/60000 [=====] - 21s 346us/step - loss: 1.0488 - acc: 0.6583 - val_loss: 0.2787 - val_acc: 0.9176  
Epoch 2/10  
60000/60000 [=====] - 16s 260us/step - loss: 0.3841 - acc: 0.8827 - val_loss: 0.1955 - val_acc: 0.9439  
Epoch 3/10  
60000/60000 [=====] - 16s 260us/step - loss: 0.3117 - acc: 0.9061 - val_loss: 0.1772 - val_acc: 0.9465  
Epoch 4/10  
60000/60000 [=====] - 16s 259us/step - loss: 0.2573 - acc: 0.9223 - val_loss: 0.1290 - val_acc: 0.9620  
Epoch 5/10  
60000/60000 [=====] - 15s 256us/step - loss: 0.2223 - acc: 0.9344 - val_loss: 0.1118 - val_acc: 0.9664  
Epoch 6/10  
60000/60000 [=====] - 15s 256us/step - loss: 0.1886 - acc: 0.9439 - val_loss: 0.0942 - val_acc: 0.9711  
Epoch 7/10  
60000/60000 [=====] - 15s 257us/step - loss: 0.1658 - acc: 0.9505 - val_loss: 0.0806 - val_acc: 0.9753  
Epoch 8/10  
60000/60000 [=====] - 15s 255us/step - loss: 0.1440 - acc: 0.9571 - val_loss: 0.0695 - val_acc: 0.9784  
Epoch 9/10  
60000/60000 [=====] - 15s 254us/step - loss: 0.1307 - acc: 0.9616 - val_loss: 0.0662 - val_acc: 0.9785  
Epoch 10/10  
60000/60000 [=====] - 15s 254us/step - loss: 0.1195 - acc: 0.9648 - val_loss: 0.0593 - val_acc: 0.9802
```

6

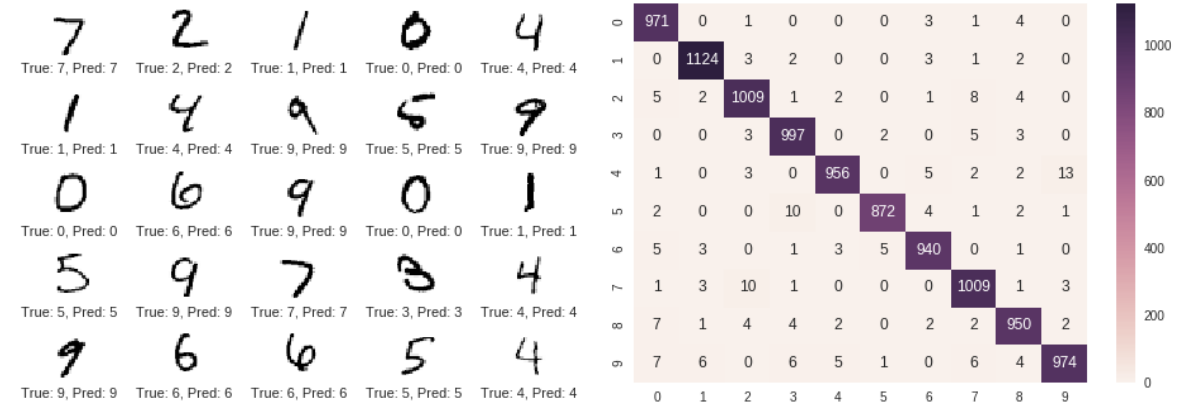
Evaluating and prediction the model

```
[ ] score = model.evaluate(x_test, y_test, verbose=0)
    print('Test loss:', score[0])
    print('Test accuracy:', score[1])
```

```
[ ] y_pred = model.predict(x_test)
```

```
[ ] plot_images(x=x_test[:25], y_true=y_test[:25], y_pred=y_pred[:25])
```

```
[ ] y_result = confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))
    sns.heatmap(pd.DataFrame(y_result, range(10), range(10)), annot=True, fmt='g')
```



CIFAR-10 Classification

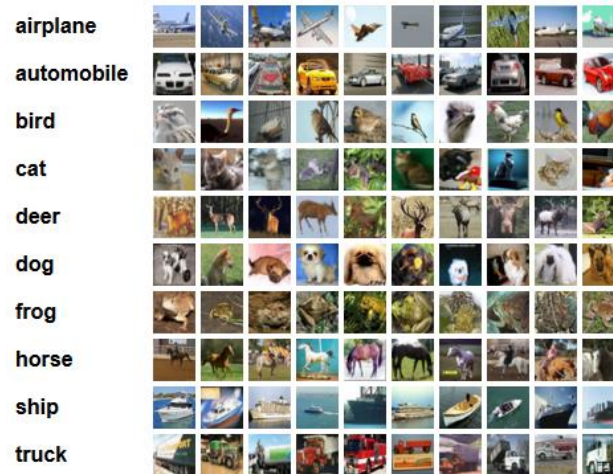
Use “05.CIFAR10.ipynb”

<https://www.cs.toronto.edu/~kriz/cifar.html>

1

CIFAR-10

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Here are the classes in the dataset, as well as 10 random images from each:



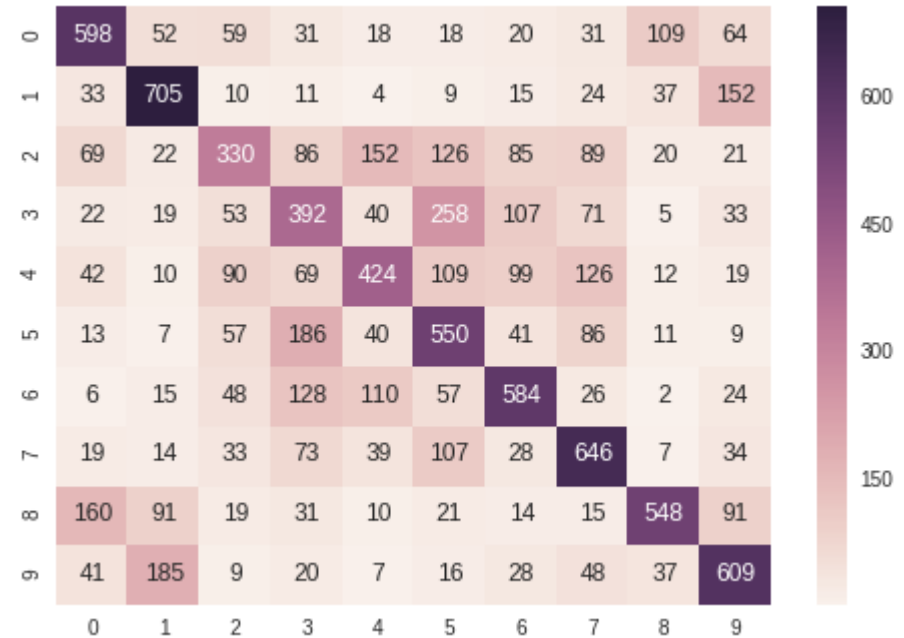
The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

2

Result of basic code

Test loss: 1.3167947158813476

Test accuracy: 0.5386



3

How to get higher accuracy

You can try the following:

- **Stack more layers.** (A key way to achieve higher accuracy!)
 - Core Layers: <https://keras.io/layers/core/>
 - Convolution Layers: <https://keras.io/layers/convolutional/>
 - Pooling Layers: <https://keras.io/layers/pooling/>
- Increase the epoch. (Do not run too many epochs.)
- Adjust the learning rate. (Between 0 to 1.)
- Use other optimizer. (<https://keras.io/optimizers/>)
- Change different activation function. (<https://keras.io/activations/>)

There are many other ways. Try to find it!

If you want to know more, go to: https://keras.io/examples/cifar10_cnn/