B455 : Project 2

> (3/3/2020) Serena Patel

> The objective of this project is to utilize the CIFAR10 dataset to create a Deep Neural Network mc color images into one of ten classes. The project will include one baseline DNN model and two a improved performace. Improved performace incldues changes in number of layers, different activ in the number of neurons in each layer, etc. Also, the optimizer utilized, the number of neurons in function applied, and the input features utilized must be detailed.

> Grading will be based on performance (accuracy) on the testing set of the baseline DNN model a Accuracy is calculated with:

```
number of correctly classified images / total number of images
```

> classes:
>
> 1. airplane
> 2. automobile
> 3. bird
> 4. cat
> 5. deer
> 6. dog
> 7. frog
> 8. horse
> 9. ship
> 10. truck

Training set: 50,000 32x32 images

Testing data: 10,000 images

Data Analysis

Deep Neural Network

A deep neural network, or DNN, is a neural network with more than one hidden layer. A DNN is a gener analysis and classification, we have to use variants of a deep neural network.

CNN

A CNN, or Convolutional Neural Network is a type of deep neural network whose input is specific to im
to this dataset.

"A convolution multiplies a matrix of pixels with a filter matrix or 'kernel' and sums up the multiplicatio
slides over to the next pixel and repeats the same process until all the image pixels have been covere

Activation Function Analysis

To find the right activation function, we have to consider our options. In particular, the sigmoid functio

> *The Sigmoid Function* This function is utilized to caputre non-linearity in the data and is used to fi
> layer with gradient descent and backpropogration. It does assume that the independent variables
> distribution, an indication of randomly occurring events with a good starting generic distribution.
> and there are problems with that. As the number of layers increases with a large input data set, th
> much it should be to show change. This can be problematic.

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

> *The ReLU Function* The Rectified Linear Unity function is a deep learning activation function that "
> negative input, but for any positive value x, it returns that value back."(Understanding Nueral Netw

Since it is influence by all x values, the ReLU function gives us more complete information, and perfor

Keras Implementation

Building a network based on the CNN model & the code from the project 2 class slides that detail a DN

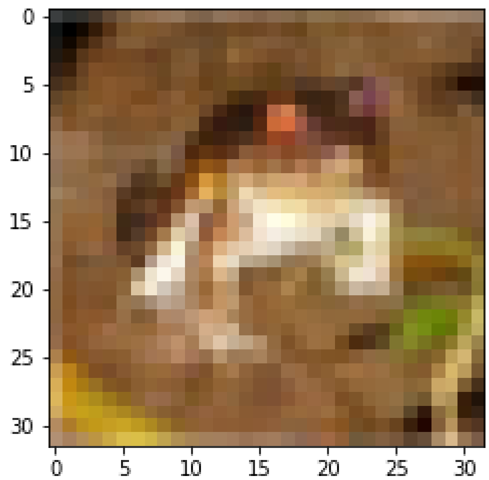
import the data

```
import numpy
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical

#load dataset
from keras.datasets import cifar10
(trainX, trainY), (testX, testY) = cifar10.load_data()
# one hot encode target values
trainY = to_categorical(trainY)
testY = to_categorical(testY)

plt.imshow(trainX[0])
trainX[0].shape
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info.

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 6s 0us/step
(32, 32, 3)
```



normalize the data

> Our data is normalized by dividing each value by the maximum color value, 255, to classify the nc

```
# convert from integers to floats
train_norm = trainX.astype('float32')
test_norm = testX.astype('float32')
# normalize to range 0-1
train_norm = train_norm / 255.0
test_norm = test_norm / 255.0
```

baseline model

> In this model, we utilize the ReLU activation function with a singular base layer. I utilized 32 becau
> the image.

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.optimizers import SGD
```

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

# compile model
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(trainX, trainY, epochs=100, batch_size=64, validation_data=(testX, testY), v

# evaluate model
_, acc = model.evaluate(testX, testY, verbose=0)
print('> %.3f' % (acc * 100.0))
```

```
10000/10000 [==============================] - 1s 83us/step
Accuracy: 0.1
```

Model 2

> In this model, the improvement made was the addidtion of the convolutional layer that doubles th
> from 32 to 64. This allows the number of output filters in the convolution to be doubled from the

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.optimizers import SGD

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

# compile model
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(trainX, trainY, epochs=100, batch_size=64, validation_data=(testX, testY), v

# evaluate model
_, acc = model.evaluate(testX, testY, verbose=0)
print('> %.3f' % (acc * 100.0))
```

```
10000/10000 [==============================] - 1s 87us/step
Accuracy: 0.1
```

Model 3

> In this model, the improvement made was the addidtion of the convolutional layer that triple the i
> 32 to 128. This allows the number of output filters in the convolution to be tripled from the origina

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.optimizers import SGD

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

# compile model
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(trainX, trainY, epochs=100, batch_size=64, validation_data=(testX, testY), v

# evaluate model
_, acc = model.evaluate(testX, testY, verbose=0)
print('> %.3f' % (acc * 100.0))
```

```
> 10.000
```

Conclusion

Our data provides us with images that are scanned, data points are extracted and classified. Since we has 10 nodes. We found that our images are square 32 x 32 images, we also found that it has 3 differ 32 to build the shapes of our images. After loading our data, we normalize it by converting our values the maximum, which is 255. 255 is the value of a full color pixel. We then are able to implement our m model. Inspired by the Keras Documenation on CNN example, I created a model that utilizes three lay the number of output filters in the convolution is 32, 64, and 128. With each new DNN, we added a nev feature detector along with the functionality to interpret the features and make predictions of classific ReLU activation function was used because it is more effiecient than a standard sigmoid function. The SGD, Stochastic gradient descent. This is a variation of the gradient descent that updates for each tra more accurate reading for each data point. The optimizer loss function utilized was the categorical cr multiple classes to consider to help watch the accuracy of our classification. The accuracies approxin be better than random guessing.

References

1. Your First Deep Learning Project in Python with Keras Step-by-Step
2. How to Configure the Number of Layers and Nodes in a Neural Network
3. ReLU Introduction
4. Understanding Neural Networks
5. Building a Convolutional Neural Network in Keras
6. Keras Documenation CNN