Images, text, video, sound and generative models

Spyros Samothrakis
Research Fellow, IADS
University of Essex

March 7, 2017

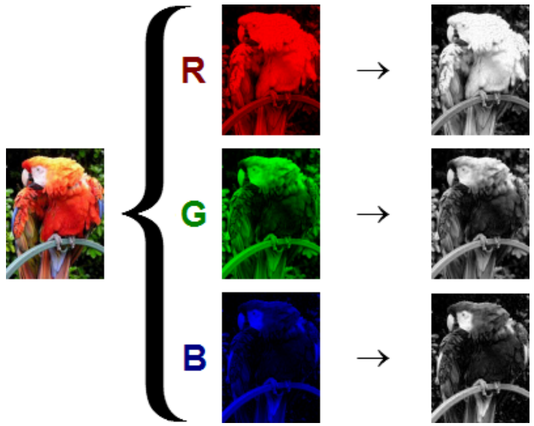
About

- We will now turn our attention on data that has less clear structure
- Sometimes called *unstructured data*
  - VS *structured* data, i.e. database like tables
- Is there anything special about this data?
  - It's the default data humans perceive and generate!
- Most machine learning benchmarks are on text or image datasets
- Neural networks excel, but there are other approaches

Image data

- Each image is composed of a number of pixels
- Pixels have different intensities
- Also, three channels (RGB)
- So in effect, we have a three dimensional structure
- Width x Height x Channels x Intensity
- 32-bit floating point numbers

RGB example



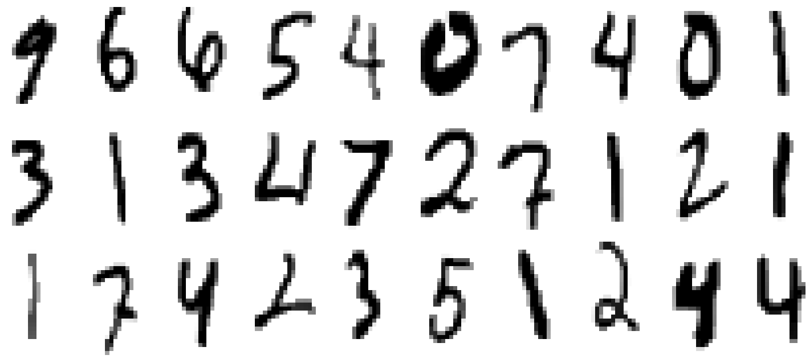
[1]

[1]http://triplelift.com/2013/07/02/the-complexity-of-image-analysis-part-2-colors/

MNIST

Very popular benchmark

60,000 training examples, 10,000 test examples, 256 different pixel values, 10 digits,



Common image preprocessing steps

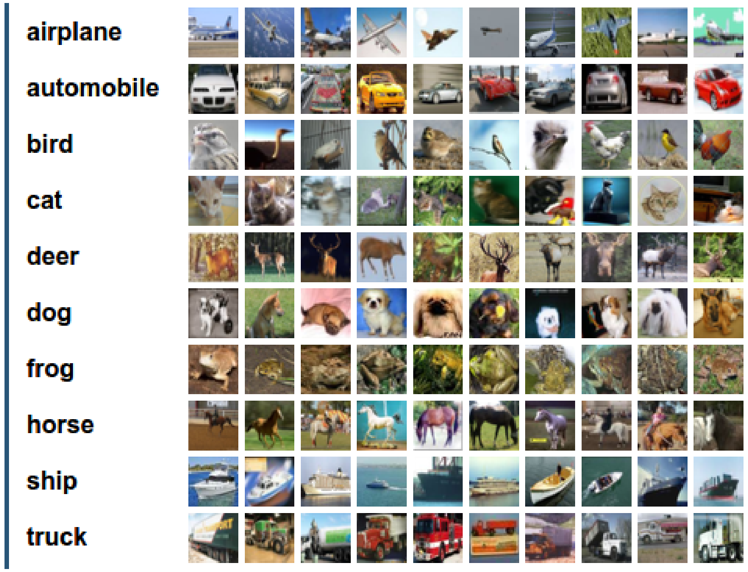
- 28*28 = 784 features
- Naive solution
  - Throw the features to a classifier/regressor
  - Subtract the mean, divide by the standard deviation
  - fit/predict
- This might not work that well

Data trumps algorithms

- It is often tempting to try to find a better algorithm to solve a certain problem
- But it has been shown time and time again that one much better off by adding more data
- Problems with neat solutions are very rare, more data
- *Physics envy* [2]
  - “An informal, incomplete grammar of the English language runs over 1,700 pages”
- We are modelling human perception as much as we are modelling cars or numbers!

[2]Halevy, Alon, Peter Norvig, and Fernando Pereira. "The unreasonable effectiveness of data." IEEE Intelligent Systems 24.2 (2009): 8-12.

CIFAR 10



Data augmentation

```
keras.preprocessing.image.ImageDataGenerator(featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    rotation_range=0.,
    width_shift_range=0.,
    height_shift_range=0.,
    shear_range=0.,
    zoom_range=0.,
    channel_shift_range=0.,
    fill_mode='nearest',
    cval=0.,
    horizontal_flip=False,
    vertical_flip=False,
    rescale=None,
    dim_ordering=K.image_dim_ordering())
```


CIFAR-10 Data Augmentation

Keras Code

```
datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True)

# compute quantities required for featurewise normalization
# (std, mean, and principal components if ZCA whitening is applied)
datagen.fit(X_train)

# fits the model on batches with real-time data augmentation:
model.fit_generator(datagen.flow(X_train, Y_train, batch_size=32),
                    samples_per_epoch=len(X_train), nb_epoch=nb_epoch)
```

Outside Keras

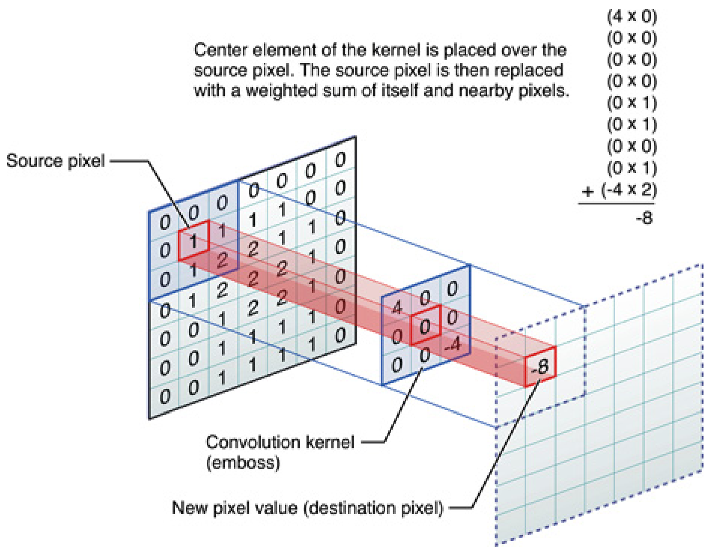
```
for i, (X_batch, Y_batch) in enumerate(datagen.flow(X_train, Y_train, batch_size=32)):
    ## break once you are happy or use an incremental regressor classifier
    ## .partial_fit
```

Can you do the same data augmentation operations on MNIST images?

Convolutional layers

- Another common approach is to constraint the number of parameters
- In a layer type in neural networks become very popular due to huge successes in computer vision
- It tries to learn different filters
    - Have you ever played with photoshop filters?

2D convolutions

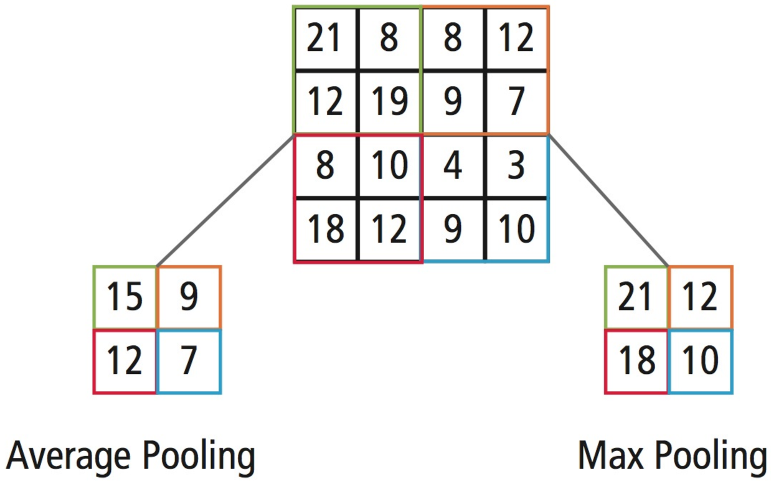


https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

Learnning 2D convolutions

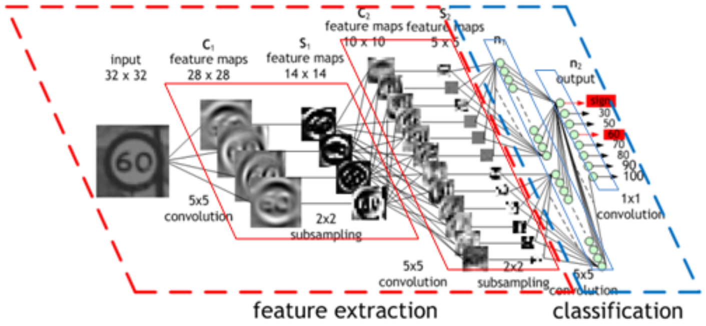
- You pass the filter over the whole image
    - Some way of treating borders
        - Padding with zeros
        - Do not calculate values if the kernel cannot fit
          http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
- Notice that now the size of the image doesn't matter as much
- 3x3 kernels very common

Pooling



http://www.embedded-vision.com/sites/default/files/technical-articles/CadenceCNN/

Visualisation



https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

All the above operations are super-optimised for GPUs

Code

```
model = Sequential()

model.add(Convolution2D(32, 3, 3, border_mode='same',
                        input_shape=X_train.shape[1:]))
model.add(Activation('relu'))
model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 3, 3, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))
```


Video data

- Video is effectively a stream of images
- It has a time component
- Multiple ways of attacking this
- You can unfold and create a really large image!
- Or, 3D convolutions!
- Not too many benchmarks
  - It's hard to annotate video
    https://www.kaggle.com/c/youtube8m/

Text

- There are multiple ways to treat text
- We will only see the ones here that require minimal pre-processing
  - You can create grammars, pre-process
  - These things are covered mostly under an NLP module
- We treat text as data

Embedding layers

- "The quick brown fox jumps over the brown lazy dog"
- Convert each to word to an integer

The	1
quick	2
brown	3
fox	4
jumps	5
over	6
the	1
brown	3
lazy	7
dog	8

Training (1)

- A weight matrix W is created as usual with size (n_words, n_neurons)
- Each row represents a word
- Each column is a specific feature/neuron
- These weights are what is passed to the follow-up layers
- What is the supervised signal?

Training (2)

- Continuous bag of words
  - You are given as input n previous words and n follow up words and you try to predict the one in the middle
- $W[\text{``}Paris\text{''}] - W[\text{``}France\text{''}] + W[\text{``}Italy\text{''}] \simeq W[\text{``}Rome\text{''}]$
- $W[\text{``}king\text{''}] - W[\text{``}man\text{''}] \simeq W[\text{``}queen\text{''}] - W[\text{``}woman\text{''}]$
- You don't need to use pre-trained vectors

Code - Preprocessing

```
print('Loading data...')
(X_train, y_train), (X_test, y_test) = imdb.load_data(nb_words=max_features)
print(len(X_train), 'train sequences')
print(len(X_test), 'test sequences')

print('Pad sequences (samples x time)')
X_train = sequence.pad_sequences(X_train, maxlen=maxlen)
X_test = sequence.pad_sequences(X_test, maxlen=maxlen)
print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
```


Code

```
model = Sequential()


model.add(Embedding(max_features,
                    embedding_dims,
                    input_length=maxlen,
                    dropout=0.2))


model.add(Convolution1D(nb_filter=nb_filter,
                        filter_length=filter_length,
                        border_mode='valid',
                        activation='relu',
                        subsample_length=1))
model.add(GlobalMaxPooling1D())

# We add a vanilla hidden layer:
model.add(Dense(hidden_dims))
model.add(Dropout(0.2))
model.add(Activation('relu'))
```


Can we do it without neural networks?

- Easy solution - create feature vector
- Very recent solution which somewhat works
  - Break the sentence/images into windowed sequences
  - i.e. generate more examples from each data point
  - Classify each of these examples
  - Combine the results into of the classifiers using a third classifier

Example

X[0] = "This film is the worst film I have ever watched. I hate the director and all the actors should be fired"

y = 1

- Window length of 4 (and obviously you need to turn words into numbers)

"This film is the", 1

"film is the worst", 1

. . . , 1

Recurrent networks

- You can use convolutions to processes sequences
- You can just flatten the sequence
- But often sequences have different length
  - You can pad
- How about arbitrary long sequences
  - E.g. a book?
  - Very long videos?
- Use a recurrent layer
  - Takes input of type (n_timesteps,n_features)

Equation

$$\mathbf{h}_t = (\mathbf{W} * \mathbf{h}_{t-1} + \mathbf{U} * x_t)$$

- h_{t-1} is the previous state
- $\mathbf{W}$ is your internal weight matrix
- $\mathbf{U}$ your external weight matrix
- x_t is the input
- Come in multiple variants - GRUs, LSTMS etc

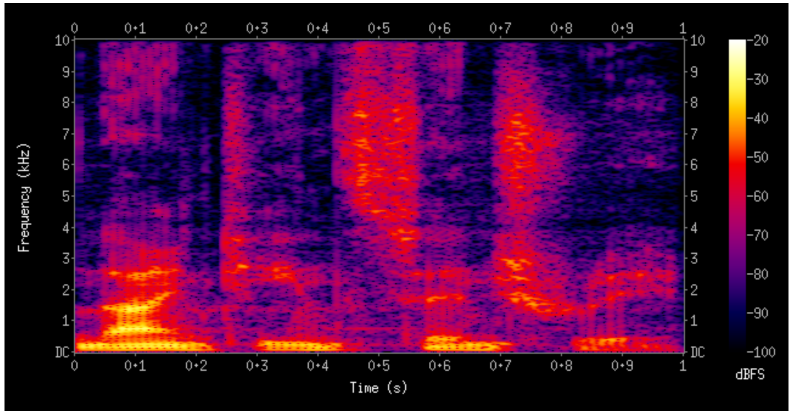
Code

```
model = Sequential()
model.add(Embedding(max_features, 128, dropout=0.2))
model.add(LSTM(128, dropout_W=0.2, dropout_U=0.2))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```


Sound

- Sound is captured in terms of
  - Bit depth (e.g. 16bit)
  - Sample size (e.g. 44.1KHz)
- 16 bits of information are collected times a 44100 second
- You could feed this directly to an RNN/Conv network
- But usually a spectrogram is passed
  - You turn audio into an image classification problem!

Spectrogram



By Aquegg - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=5544473
https://yerevann.github.io/2016/06/26/combining-cnn-and-rnn-for-spoken-language-identification/

Autoencoders

- Your goal is to learn the data
- There is no other supervisory signal, but the data
- I'll give you an image as X
  - You will produce the same image as output
- Applications?
  - Image de-noising
  - Compression!

Example code

```
# this is the size of our encoded representations
encoding_dim = 32

# this is our input placeholder
input_img = Input(shape=(784,))
encoded = Dense(encoding_dim, activation='relu')(input_img)
decoded = Dense(784, activation='sigmoid')(encoded)
autoencoder = Model(input=input_img, output=decoded)
```


Generating text

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

- Pushed the popularity of generative methods sky-high
- Learn to generate text, given some examples

Example

- Try to predict characters one by one
- You input a character
  - You call .fit
  - Network is stateful
    - i.e. it remembers where you left off!
- This way you can process super-long sequences iteratively

Code

```
keras.layers.recurrent.Recurrent(weights=None,
                                 return_sequences=False,
                                 go_backwards=False,
                                 stateful=False,
                                 unroll=False,
                                 consume_less='cpu',
                                 input_dim=None,
                                 input_length=None)
```


GANs

- Claims of being the most important advance of in AI for years
- Define a game of sorts
  - One network G generates an image/text/video
  - Another network D tries to discriminate between real and artificial examples!
  - G is trained as to produce images that D cannot differentiate! `https://www.youtube.com/watch?v=PmC6ZOaCAOs&feature=youtu.be`

Conclusion

- There is more to data than just tables
- Arguably, the most interesting data is in a table format
- Again, we have just touched upon the subject
- What about sound?
  - Wavenet