Deep Learning with Keras

Ing. Rodolfo Bonnin.

Contents

- **Geting started**
- **Multi-layer Feed Forward Networks**: We will explore the basic elements of the Keras api, especially the Sequential Object.
- **Layer exploration**: We will use the Keras api to explore a determined model.
- **Convolutional Networks**:We will learn how to build CNN (Convolutional Neural Networks), using Convolutional, Pooling and DropOut layers
- **Transfer Learning**: Will discover the Keras implementation of some foundational Deep Convolutional architecures, and use transfer learning to apply them to new problems.
- **Neural network visualization with Quiver**: We will explore the layer of complex neural networks, and graphically analyze the different stages in the succesive stacked layers.
- **Recursive Neural Networks**: We will use keras.layers.convolutional_recurrent to predict values of a time series.
-

Who am I?

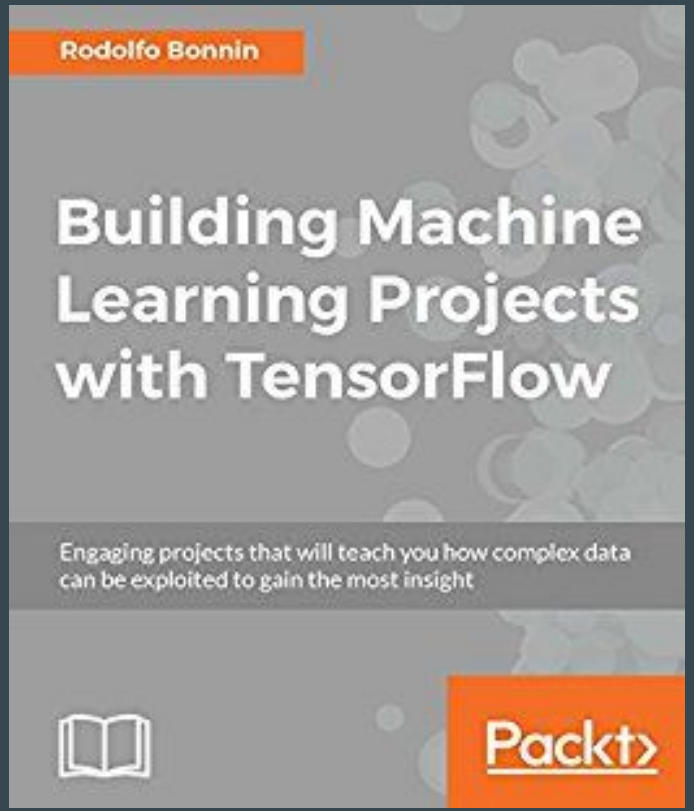
Systems Engineer

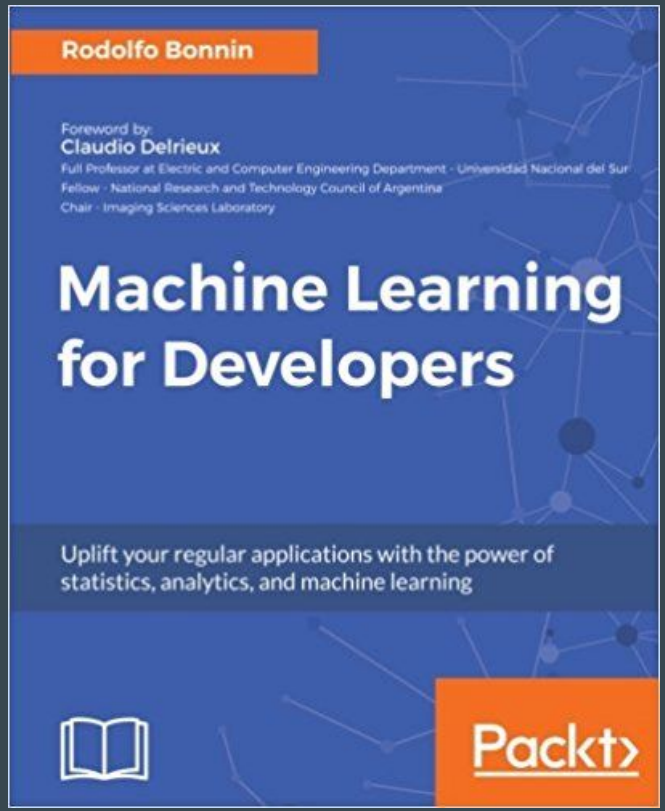
Ph.D Student, UTN

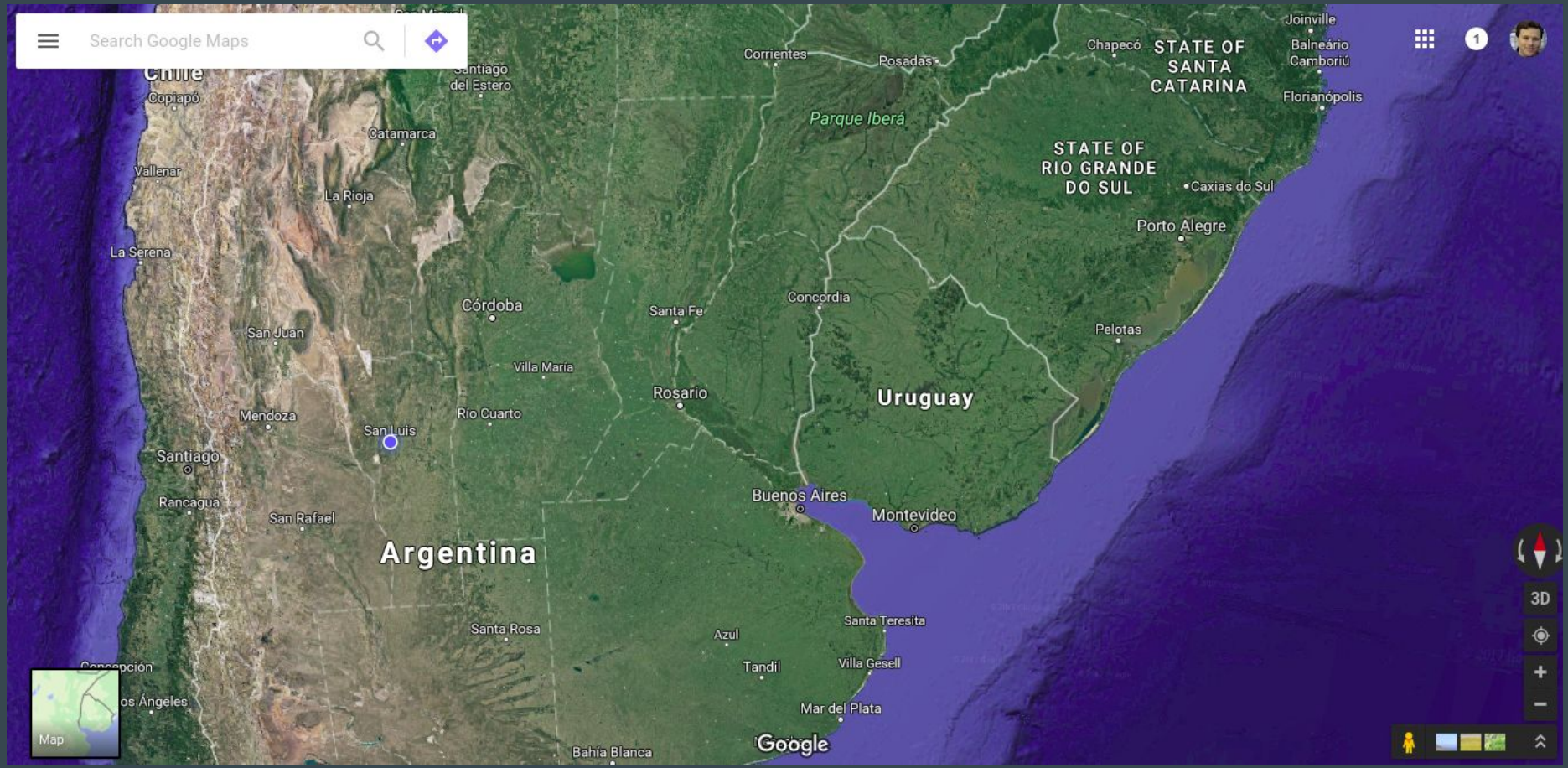
Started working on Convolutional Neural Networks on 2008 (Before the boom)

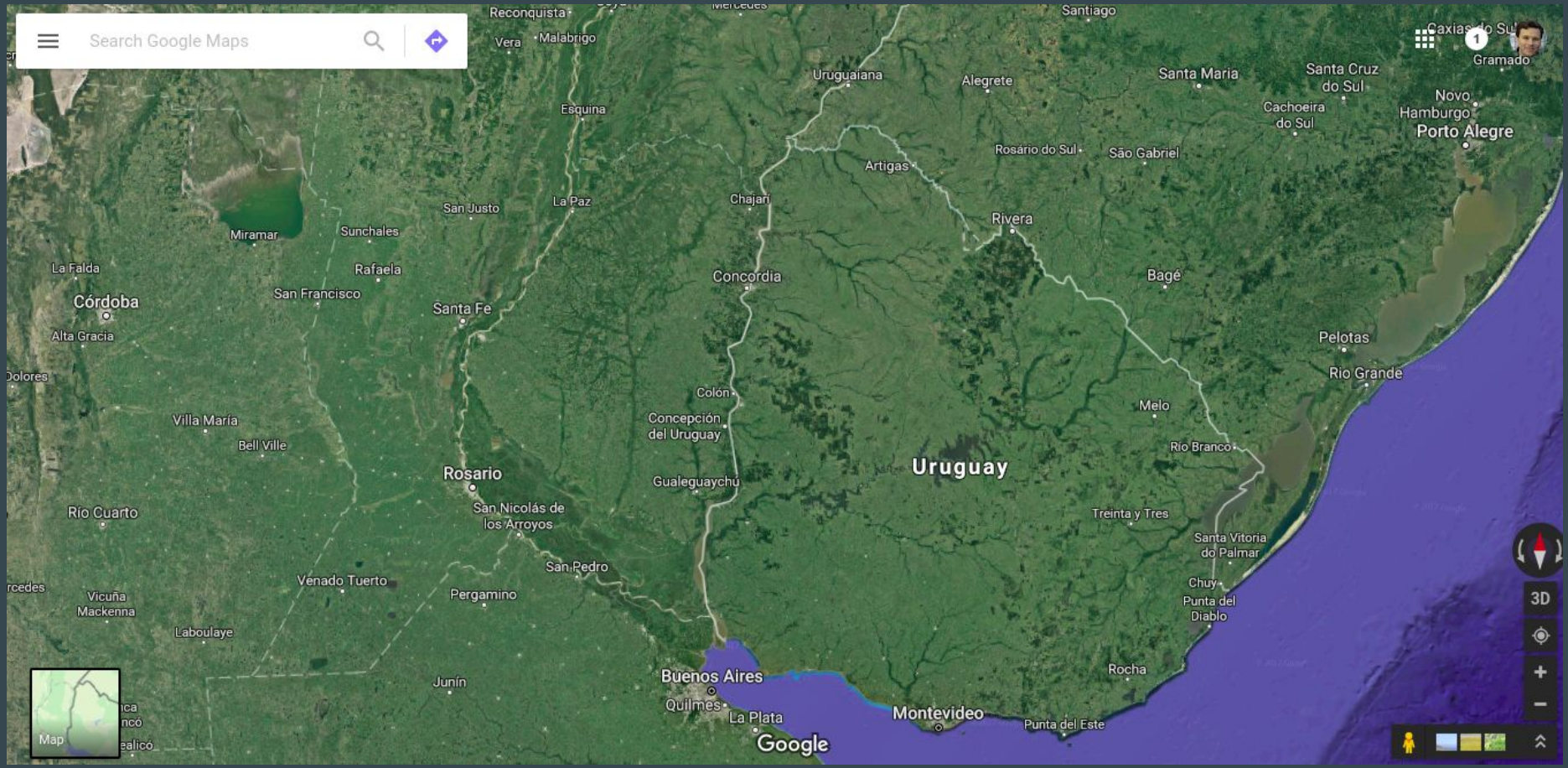
ML @ Mercadolibre

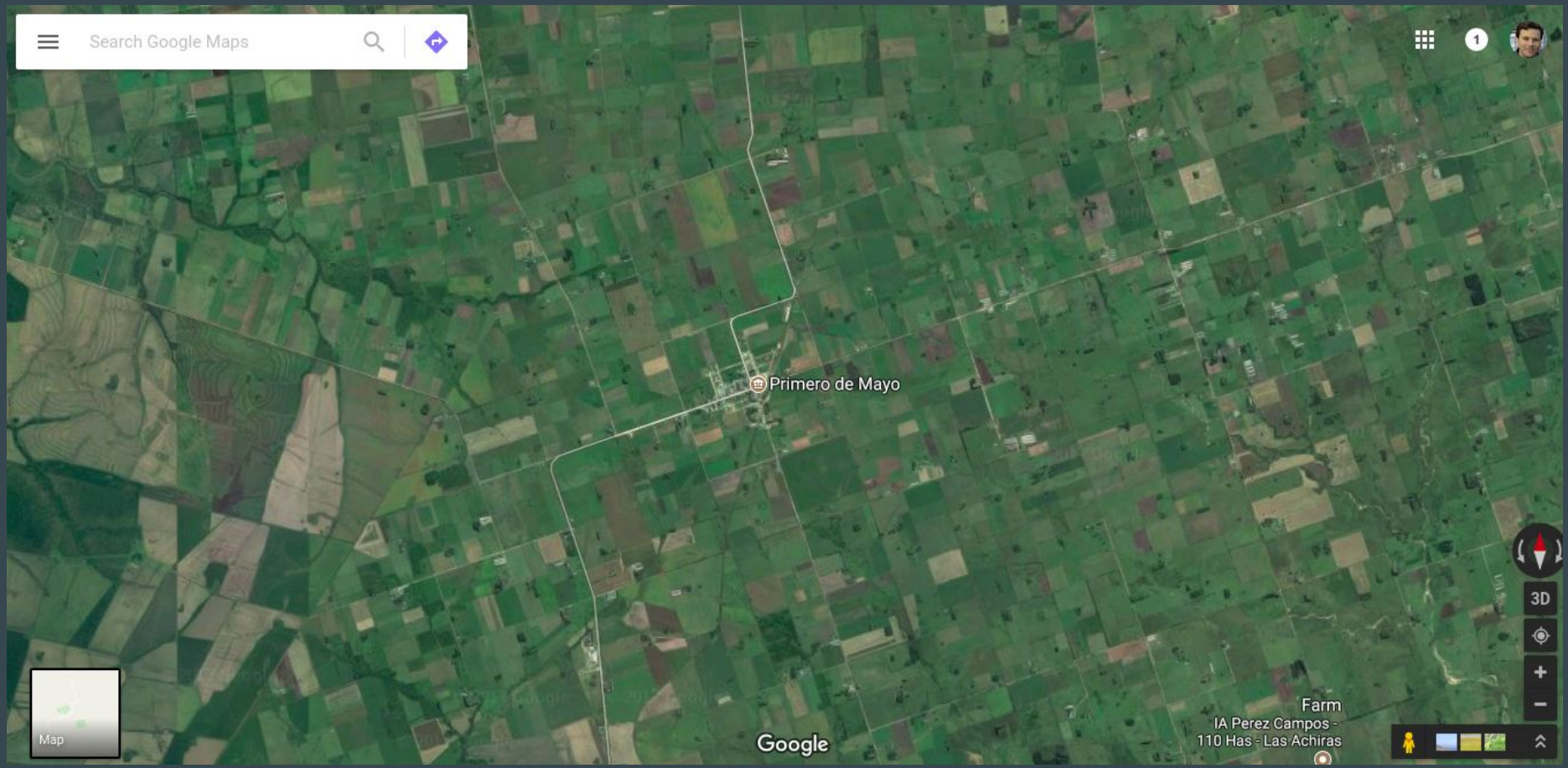
Work @ Machinalis






Search Google Maps
Chile
Copiapó
Catamarca
Vallenar
La Rioja
La Serena
Córdoba
San Juan
Villa María
Mendoza
Río Cuarto
San Luis
Santiago
Rancagua
San Rafael
Argentina
Santa Rosa
Concepción
Los Ángeles
Bahía Blanca
Corrientes
Posadas
Santiago del Estero
Parque Iberá
Chapecó
STATE OF SANTA CATARINA
Joinville
Balneário Camboriú
Florianópolis
STATE OF RIO GRANDE DO SUL
Caxias do Sul
Porto Alegre
Pelotas
Concordia
Santa Fe
Rosario
Uruguay
Buenos Aires
Montevideo
Santa Teresita
Azul
Tandil
Villa Gesell
Mar del Plata
Google
Map
3D

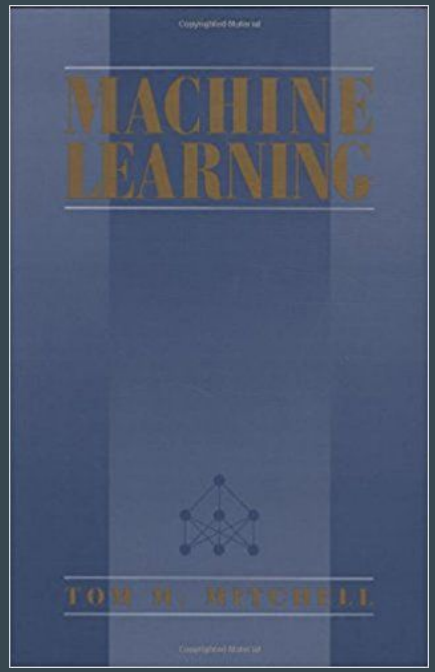

Search Google Maps
Reconquista
Vera
Malabrigo
Santiago
Uruguaiana
Alegrete
Santa Maria
Santa Cruz do Sul
Cachoeira do Sul
Novo Hamburgo
Gramado
Porto Alegre
Esquina
Rosário do Sul
São Gabriel
Artigas
Chajarí
La Paz
San Justo
Miramar
Sunchales
Rivera
La Falda
Rafaela
Concordia
Bagé
Córdoba
San Francisco
Santa Fe
Alta Gracia
Pelotas
Rio Grande
Colón
Concepción del Uruguay
Melo
Villa María
Bell Ville
Río Branco
Uruguay
Rosario
Gualeguaychú
Río Cuarto
San Nicolás de los Arroyos
Treinta y Tres
Santa Vitoria do Palmar
San Pedro
Venado Tuerto
Chuy
Pergamino
Punta del Diablo
Vicuña Mackenna
Laboulaye
Rocha
Junín
Buenos Aires
Quilmes
La Plata
Montevideo
Punta del Este
Google
Map
3D


Search Google Maps
Primero de Mayo
Farm
IA Perez Campos -
110 Has - Las Achiras
Google
Map
3D

Machine Learning: definitions

“The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.”

Tom Mitchell



Machine Learning: definitions

“ A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”

Tom Mitchell

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two,)and others via complementary projects)
- Runs seamlessly on CPU and GPU.

Core Models: Sequential

The Sequential model is a linear stack of layers.

3 main Steps:

1. Specify the input shape
2. compile()
3. fit()

Input shape

Model compilation

It need 3 elements

- An optimizer. This could be the string identifier of an existing optimizer (such as rmsprop or adagrad), or an instance of the Optimizer class.
- A loss function. This is the objective that the model will try to minimize. It can be the string identifier of an existing loss function (such as categorical_crossentropy or mse), or it can be an objective function.
- A list of metrics.

```
model.compile(loss='mean_squared_error',
              optimizer='sgd',
              metrics=[metrics.mae,
metrics.categorical_accuracy])
```

Model Training

Keras models are trained on Numpy arrays of input data and labels.

For training a model, you will typically use the `fit` function.

Layer Types: Core layers

Just your regular densely-connected NN layer.

```
Activation:Applies an activation function to an output.
Dropout:Applies Dropout to the input.
Flatten:Flattens the input. Does not affect the batch size.
Reshape
Permute:Permutes the dimensions of the input according to a
given pattern.
RepeatVector
Lambda
```

Layer Types: Convolutional layers

`Conv1D`

`Conv2D`

`Conv3D`

Layer Types: Pooling layers

```
MaxPooling1D  (& 2D & 3D)

AveragePooling1D (& 2D & 3D)
```

Layer Types: Recurrent layers

`RNN`
`SimpleRNN`
`GRU`
`LSTM`
`ConvLSTM2D`
`StackedRNNCells`

Specially fast Convolutional layers:
`CuDNNGRU`
`CuDNNLSTM`

Text utilities

`Text_to_word_sequence:` Split a sentence into a list of words.

`One_hot:` Encodes words into an index

`Tokenizer:` Class for vectorizing texts, or/and turning texts into sequences (=list of word indexes, where the word of rank i in the dataset (starting at 1) has index i).

Losses:

```
mean_squared_error

mean_squared_error(y_true, y_pred)

mean_absolute_error(y_true, y_pred)

mean_absolute_percentage_error(y_true, y_pred)

mean_squared_logarithmic_error(y_true, y_pred)

categorical_crossentropy(y_true, y_pred)
```

Metrics

```
accuracy

binary_accuracy

binary_accuracy(y_true, y_pred)

categorical_accuracy(y_true, y_pred)

top_k_categorical_accuracy(y_true, y_pred, k=5)
```

Metrics

Accuracy

binary_accuracy

binary_accuracy(y_true, y_pred)

categorical_accuracy(y_true, y_pred)

top_k_categorical_accuracy(y_true, y_pred, k=5)

Optimizers

```
keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```

```
keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)
```

```
keras.optimizers.Adagrad(lr=0.01, epsilon=1e-08, decay=0.0)
```

```
keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=1e-08, decay=0.0)
```

```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
```

Activations

```
softmax(x, axis=-1)
relu(x, alpha=0.0, max_value=None)
tanh(x)
sigmoid(x)
hard_sigmoid(x)
linear(x)
```

Example:

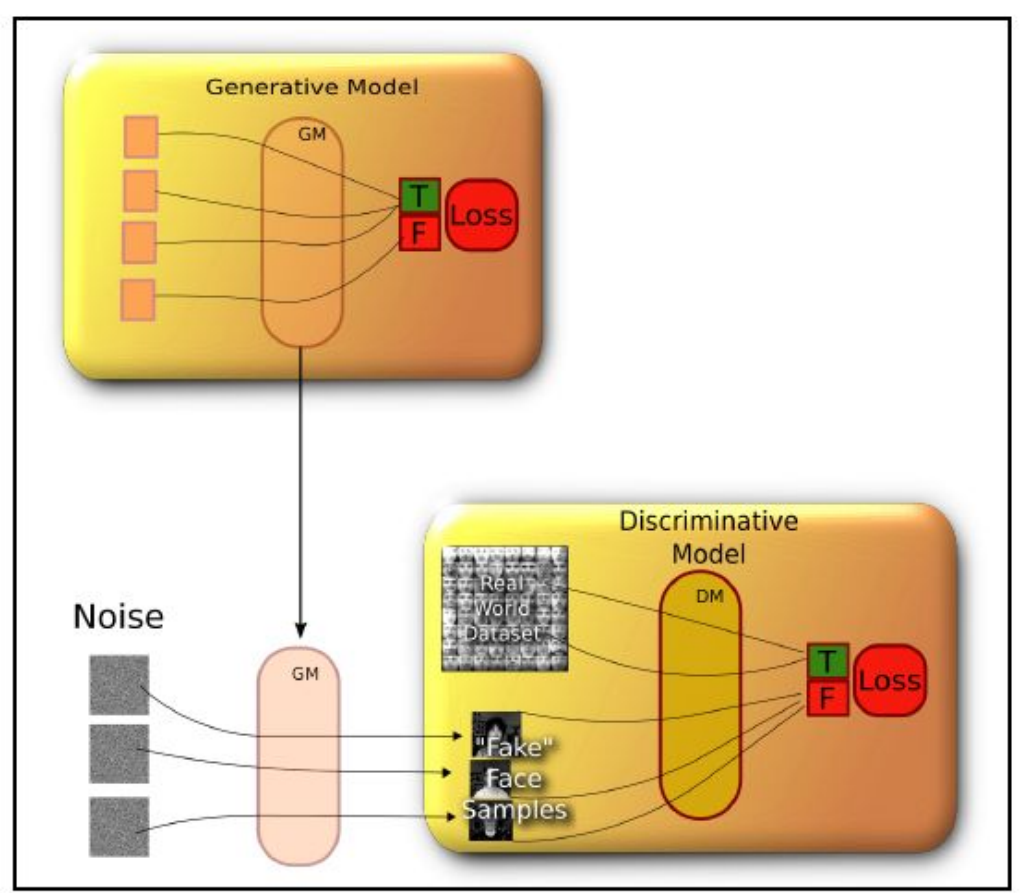
```
from keras.layers import Activation, Dense
model.add(Dense(64))
model.add(Activation('tanh'))
```

The history of Neural models

Neural network visualization with Quiver

Recursive Neural Networks: example with Time Series

Generative Adversarial Networks



(a) Church outdoor.

(b) Dining room.

(c) Kitchen.

(d) Conference room.

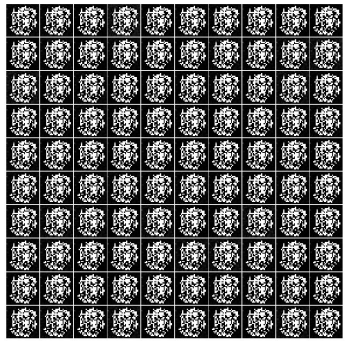
Generative Adversarial Network: Creating clothes.

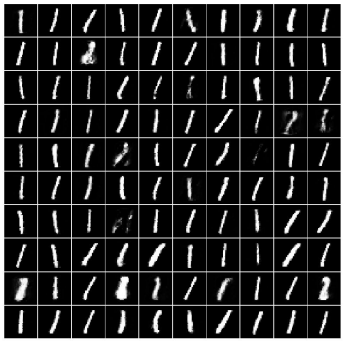
We will train a Generative Adversarial Network, using the initial dataset as starting point, and then we will generate a set of samples from random noise.

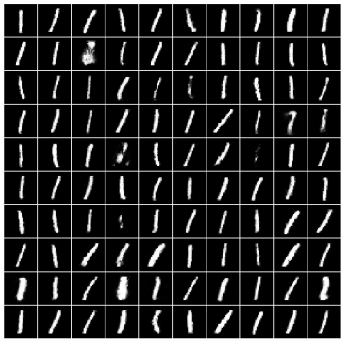
For this, we will use the keras-rl project:

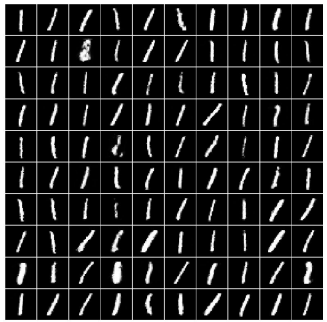
`https://github.com/matthiasplappert/keras-rl`

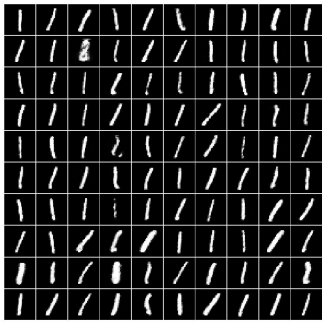
Generative Adversarial Network: GAN Results with mnist (poor)

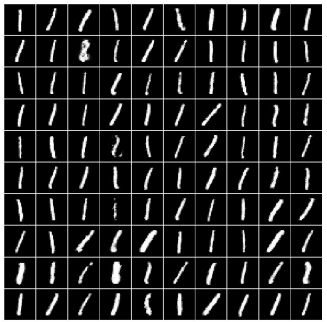












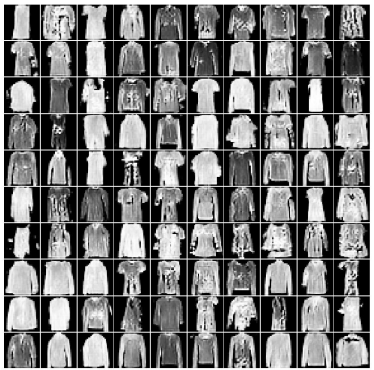
Fashion Mnist

Generative Adversarial Network: DCGANs Results

Epoch 1

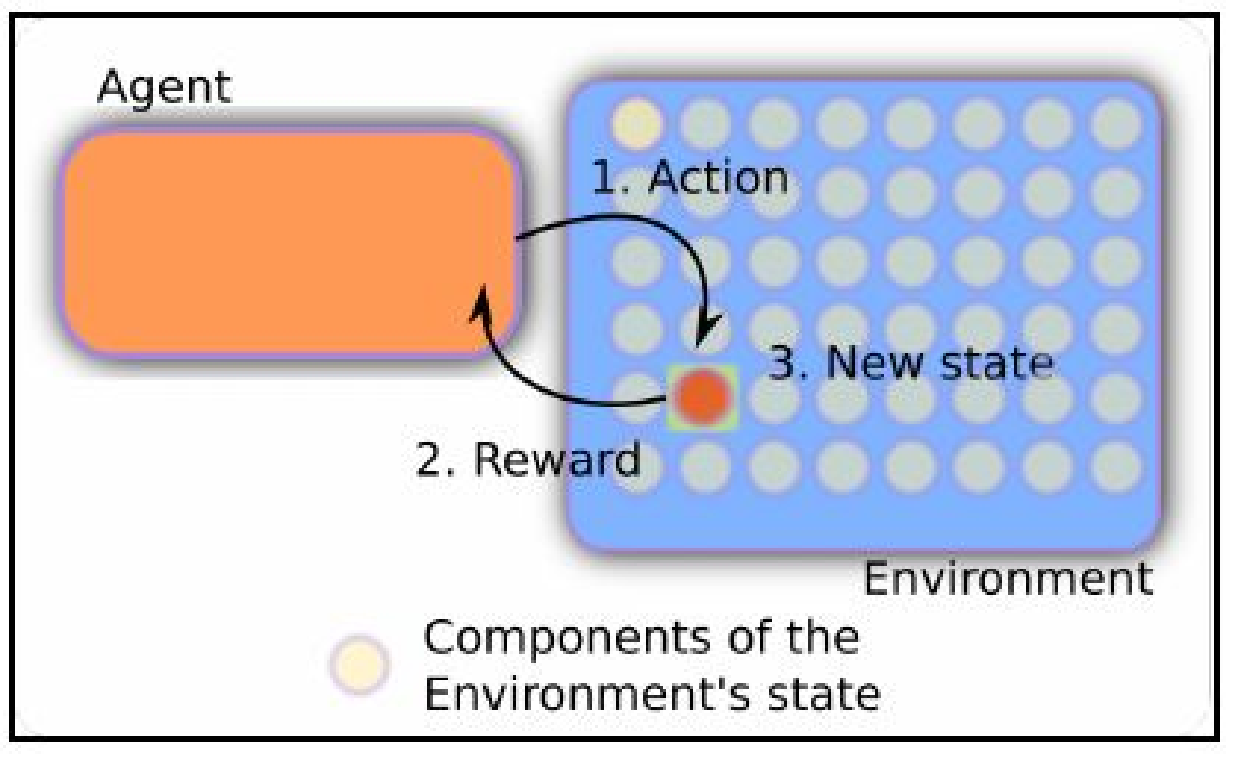
Epoch 2

Epoch 3

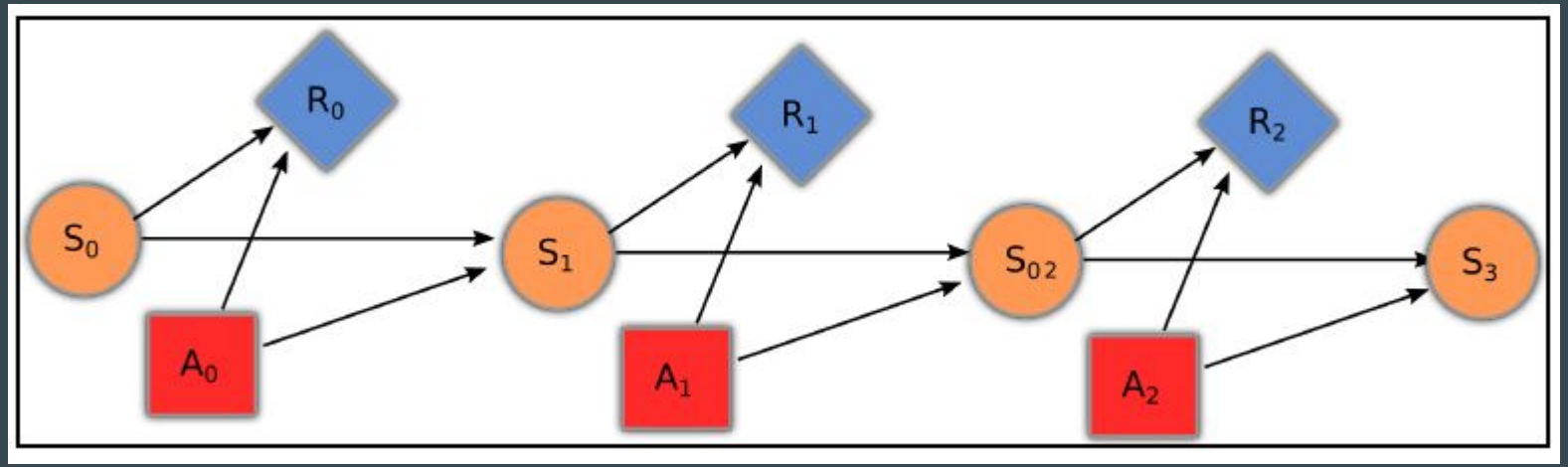


Epoch 4

Reinforcement Learning



Reinforcement Learning



Cartpole: Simple example

```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
         print (reward, done, info)
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break
```

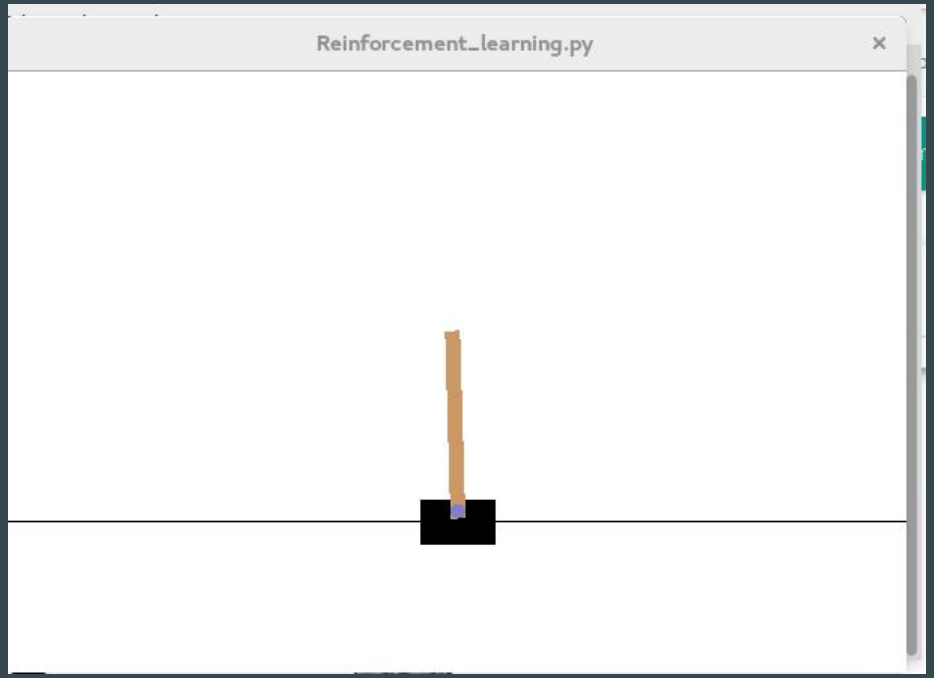
OpenAI Gym: Cartpole

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.

CartPole-v0 defines "solving" as getting average reward of 195.0 over 100 consecutive trials.

This environment corresponds to the version of the cart-pole problem described by Barto, Sutton, and Anderson.

OpenAI Gym: Cartpole



Noisy Cross-Entropy Method