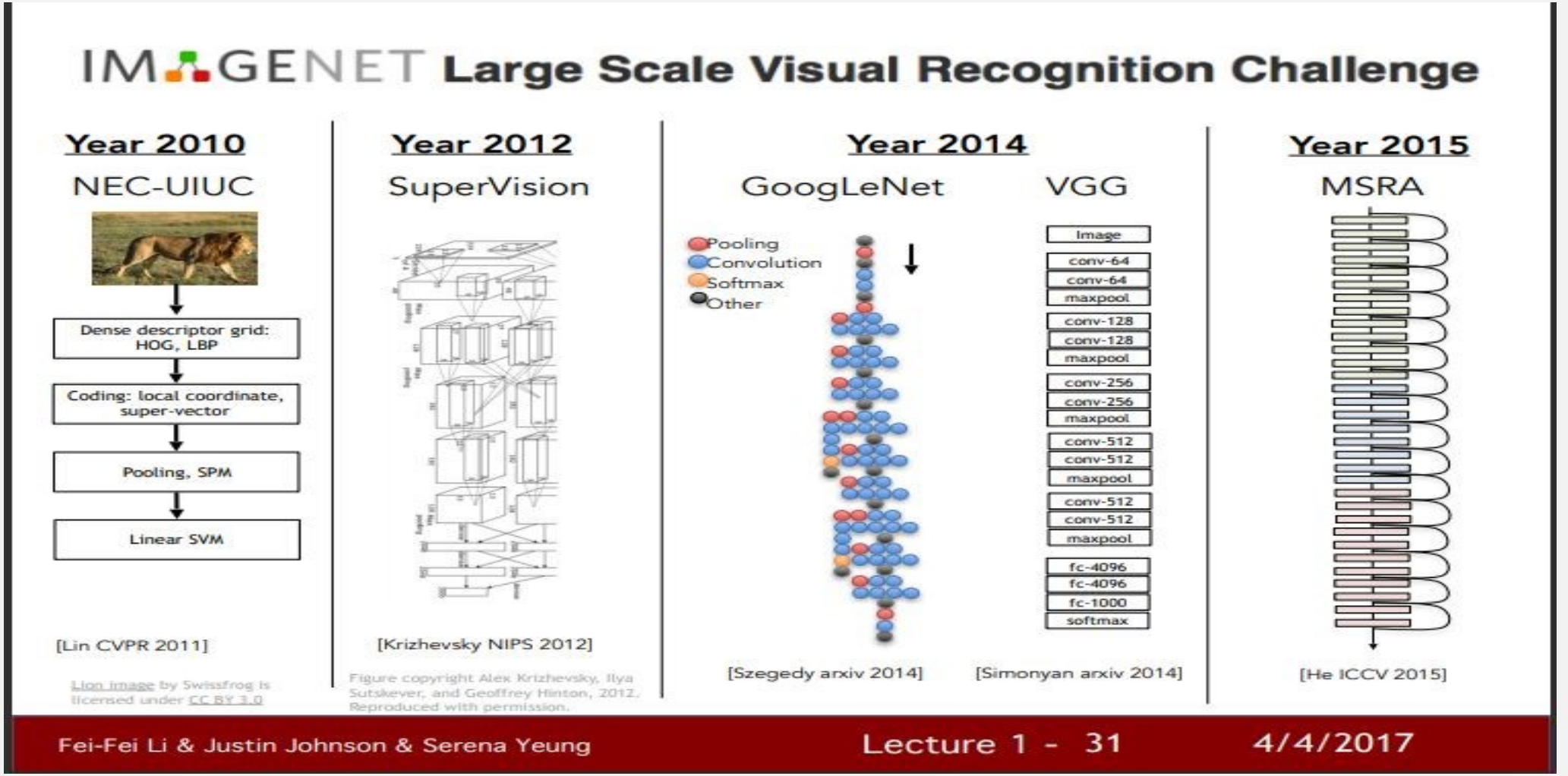
INTRODUCTION TO KERAS TUNER WITH TENSORFLOW

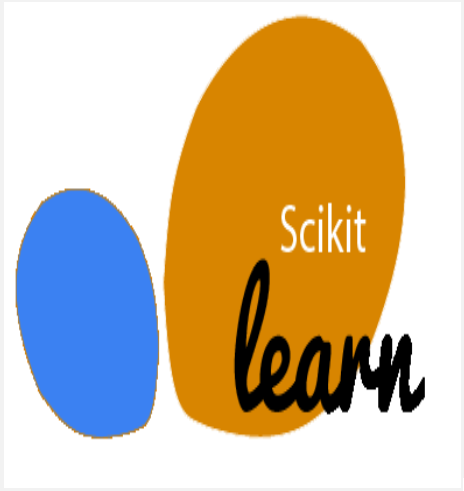
- Ashwin Phadke
AI and Deep learning Engineer | Mentor | Trainer

CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks has helped many, to breakthrough in solve the image classification challenge



Most popular frameworks for data science



UNDERSTANDING AN IMAGE

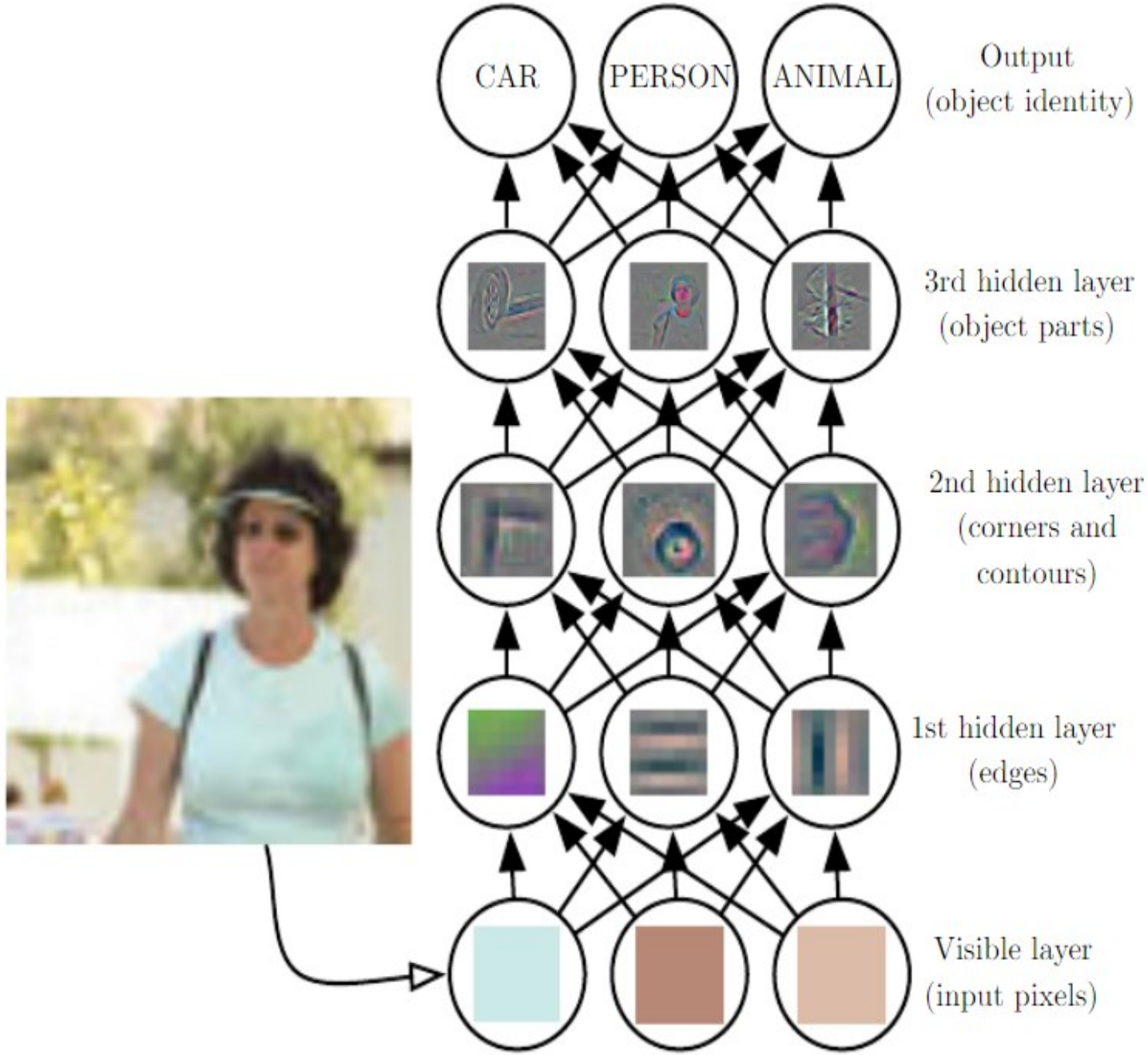
Understanding the image is possibly the extremely important task that needs to be solved. No matter how the object to be classified appears it still is that same image and needs to be classified it that way. Issues like camera angle variation, illumination, deformation, occlusion, image variations.

Cat

Still a cat

They do come in all variations, don't they

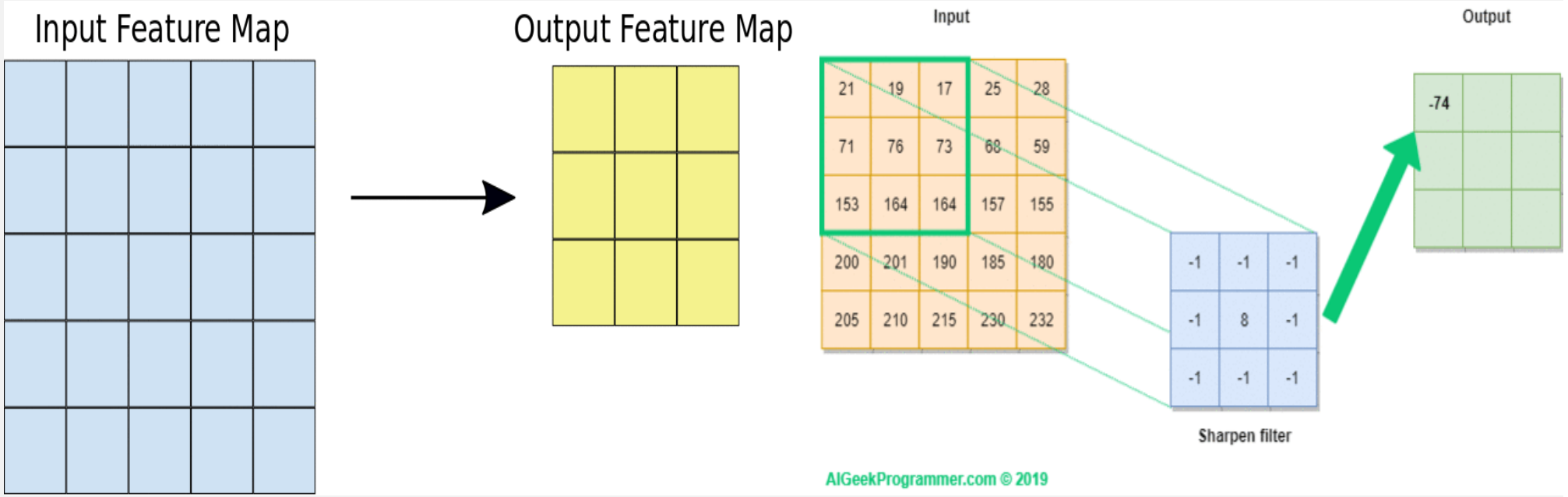
- Pixabay



Deep Learning – Ian Goodfellow, Yoshua Bengio, et.al

CONVOLUTIONAL NEURAL NETWORKS – CONVOLUTION OPERATION

Let's go ahead and know the convolutional operation ourseleves.

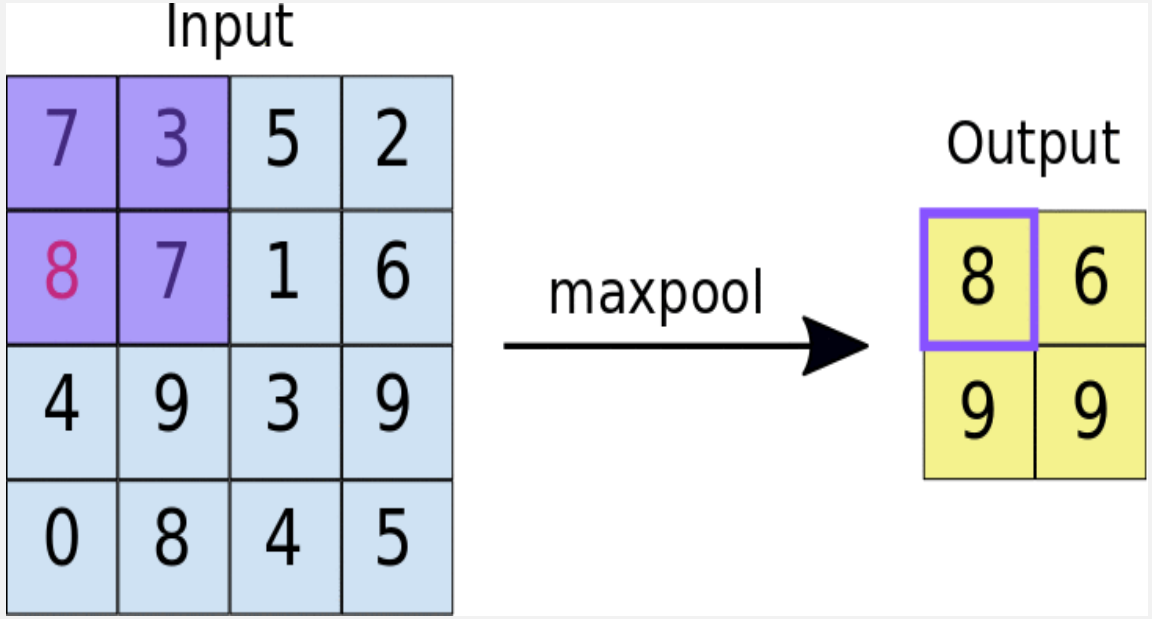


- Representations:
  - Blue : Input to the convolutional layer. (5 X 5)
  - Red : Filter being applied over the input. (3 X 3). These are learned filters. Like sharpen filter etc. Different size filter different size features.
  - Yellow : Resulting convolutional output. (3 X 3)

- ML Practicum
- AIGeekProgrammer

CONVOLUTIONAL NEURAL NETWORKS – MAX POOLING

Well the earlier we convolved over the image and scanned through each piixel



- Why use max ppoling?
  - Faster to compute as reduced computations.
  - Maximum pixel value stored hence sense or meaning of the image doesn't necessarily change that much.
  - Applied as 2X2 with stride of 2.
  - Average pooling and Max pooling.

CONVOLUTIONAL NEURAL NETWORKS – STRIDES AND PADDING

We have done convolution and also formed a simplified computation using max pool, but do we directly apply these operations over the image?

- Strides :
  - The amount of movement between applications of the filter to the input ,and it is almost always symmetrical in height and width dimensions.

- Padding :
  - Filter starts at left
  - No surrounding pixels.
  - Pad values to add for border pixel values so they are at the center.
  - padding='same' makes sure output shape is same as input.

```
model.add(Conv2D(1, (3,3), strides=(2, 2), input_shape=(8, 8, 1)))

model.add(Conv2D(1, (3,3), padding='same', input_shape=(8, 8, 1)))
```

OKAY SO HOW DO WE DO IT?

FULL VERSION – OH YEAH!

```
]  1 # TF Documentation : Sequential groups a linear stack of layers into a tf.keras.Models
   2 model = tf.keras.models.Sequential([
   3   # Generate 64 filters of size 3X3
   4   # Relu activation means negative values go away
   5   # Input shape on the input, 1 is using single byte for color as images are grayscale
   6   # Take a look at input shape and output shape , no batch_size so None - (batch_size, height, width, depth)
   7   tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(28, 28, 1)),
   8
   9   # Max pooling as we will take maximum value which is a 2X2 poll so wvery 4 pixels go to 1
  10   tf.keras.layers.MaxPooling2D(2, 2),
  11
  12   # Another layer
  13   tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
  14   tf.keras.layers.MaxPooling2D(2,2),
  15
  16   # Converts the input to 1D set instead of the square we saw earlier
  17   tf.keras.layers.Flatten(),
  18
  19   # Adds a layer of neurons
  20   tf.keras.layers.Dense(128, activation='relu'),
  21
  22   # The last layers have specific number of neurons, ask me why! :)
  23   # Softmax helps you take the largest value and convert it to 1, again is it max pool? No? Then ask.
  24
  25   tf.keras.layers.Dense(10, activation='softmax')
  26 ])
  27
```

THAT'S A LOT OF PARAMETERS TO TUNE, TRY, TEST, TEST AGAIN , HERE WE GO!

Well, gee we have keras-tuner

(There are a lot of other similar packages for pytorch, scikit-learn models here we discuss keras-tuner only.)

Before we start :
The number of trees in a random forest is a hyperparameter while the weights in a neural network are model parameters learned during training and **find the model hyperparameters that yield the best score on the validation set metric**.

KERAS TUNER

Keras Tuner is an easy-to-use, distributable hyper parameter optimization framework that solves the pain points of performing a hyper parameter search. Keras Tuner makes it easy to define a search space and leverage included algorithms to find the best hyper parameter values.

- Algorithms available with keras tuner :
  - Bayesian Optimization.
  - Hyperband
  - Random search
- Pre-made Tunable applications include :
  - HyperResNet
  - HyperXception
- Offers distributed training using `tf.distribute.Strategy`
- Custom training Loops :
  - The `kerastuner.Tuner` class can be sub classed to support advanced uses such as:
    - Custom training loops (GANs, reinforcement learning, etc.)
    - Adding hyper parameters outside of the model building function (preprocessing, data augmentation, test time augmentation, etc.)

```
export KERASTUNER_TUNER_ID="chief"
export KERASTUNER_ORACLE_IP="127.0.0.1"
export KERASTUNER_ORACLE_PORT="8000"
python run_my_search.py
```

WHAT HYPERPARAMETERS CAN YOU TUNE?

- Learning Rate [e.g 0.1, 0.01, 0.001]
- Momentum [e.g 0.0, 0.2, 0.4, 0.6]
- Number of units in an hidden layer [e.g min = 32 , max = 512]
- Number of Hidden Layers [e.g 1 to 4]
- Activation function [e.g tanh, relu, sigmoid, softmax]
- Droupout(Regularization technique) Rate [e.g 0.0, 0.1, 0.2]

Geez, That's a whole bunch of manual time saved! Imagine doing combinations of these.

To know more and understand about these hyperparameters refer to good ol Wikipeida.

KERAS TUNER BUILT IN ALGORITHMS

Bayesian Optimization
(Wikipedia)

- Build a probability model of the objective function and use it to select the most promising hyperparameters to evaluate in the true objective function.
- By evaluating hyperparameters that appear more promising from past results, Bayesian methods can find better model settings than random search in fewer iterations.
- Bayesian Optimization is an approach that uses Bayes Theorem to direct the search in order to find the minimum or maximum of an objective function.

Hyperband

- Hyperband is a variation of random search, but with some explore-exploit theory to find the best time allocation for each of the configurations.
- Randomly sample x hyperparameters.Evaluate validation loss.
- Remove lowest performers.
- Again iterate and evaluate.
- Remove low performers

@criteo-labs

Random Search

- Set up a grid of hyperparameter values and select *random* combinations to train the model and score. The number of search iterations is set based on time/resources.
- **Random search** generates random values for each hyperparameter being tested, and then uses cross validation to test the accuracy of each combination. (@jackstalfort)

KERAS TUNER – HELLO WORLD

```
from kerastuner.applications import HyperResNet
from kerastuner.tuners import Hyperband

hypermodel = HyperResNet(input_shape=(128, 128, 3), classes=10)

tuner = Hyperband(
    hypermodel,
    objective='val_accuracy',
    max_epochs=20,
    directory='my_dir',
    project_name='hello_world')

tuner.search(x, y, epochs=20, validation_data=(val_x, val_y))
```

MNIST hypermodel is as easy as 1,2,3

1. Wrap model in a function

2. Define hyper-parameters

3. Replace static value with hyper-parameters

```
def model_fn():

    LR = Choice('learning_rate', [0.001, 0.0005, 0.0001], group='optimizer')
    DROPOUT_RATE = Linear('dropout_rate', 0.0, 0.5, 5, group='dense')
    NUM_DIMS = Range('num_dims', 8, 32, 8, group='dense')
    NUM_LAYERS = Range('num_layers', 1, 3, group='dense')
    L2_NUM_FILTERS = Range('l2_num_filters', 8, 64, 8, group='cnn')
    L1_NUM_FILTERS = Range('l1_num_filters', 8, 64, 8, group='cnn')

    model = Sequential()
    model.add(Conv2D(L1_NUM_FILTERS, kernel_size=(3, 3), activation='relu'))
    model.add(Conv2D(L2_NUM_FILTERS, kernel_size=(3, 3), activation='relu'))
    model.add(Flatten())
    for _ in range(NUM_LAYERS):
      model.add(Dense(NUM_DIMS, activation='relu'))
      model.add(Dropout(DROPOUT_RATE))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer=Adam(LR))
    return model
```

Machine Learning

Deep Learning

Computer Vision

NLP

SOME HANDS ON

Alright! Get your laptops out and be quick.

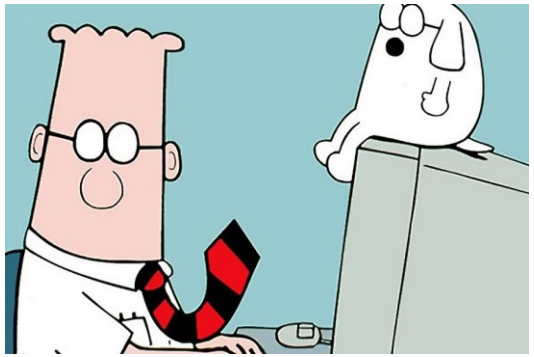
Colab Notebook:

- Simple example
- Hyperband using keras-tuner.
- Colab notebook

REFERENCES :

Here are references that helped me a lot and you should give it a try if I may suggest.

- Deep Learning book : https://deeplearningbook.org, Goodfellow et.al *
- Tensorflow Documentation - Tuner
- https://keras-team.github.io/keras-tuner/
- https://machinelearningmastery.com/
- Andrej Kaarpathy blog : http://karpathy.github.io/
- https://medium.com/@jackstalfort/hyperparameter-tuning-using-grid-search-and-random-search-f8750a464b35
- https://medium.com/criteo-labs/hyper-parameter-optimization-algorithms-2fe447525903
- Coursera : Neural netowrks, Introduction to Tensorflow, Laurence Moroney
- People to follow on LinkedIn : Dat Tran, Jason Mayes, Yogesh Kulkarni, Rahee Agate-Walambe, Usha Rangarajan, Laurence Moroney
- Powerpoint Template : Powerpoint template store.
- Google image and GIF search.



THANK YOU