ENGLISH CHARACTER IMAGE GENERATOR IMAGE PREDICTOR

ALAN WONG

CONTENTS

1 PROBLEM DEFINITION

CREATE A PROGRAMME THAT FULFILS THE FOLLOWING CRITERIA

- ENGLISH CHARACTER GENERATOR
- ENGLISH CHARACTER PREDICTOR
- DEEP NEURAL NETWORK
- PYTHON 3

2

INTRODUCTION TO DEEP LEARNING

MACHINE LEARNING
TASK TYPES

SUPERVISED LEARNING

INPUT AND OUTPUT DATA

MAKE PREDICTION

UNSUPERVISED LEARNING

INPUT DATA ONLY

FIND SIMILARITY AND DIFFERENCES IN DATA

REINFORCEMENT LEARNING

INPUT DATA ONLY

FIND ACTION TO MAXIMIZE REWARD

2

INTRODUCTION TO DEEP LEARNING

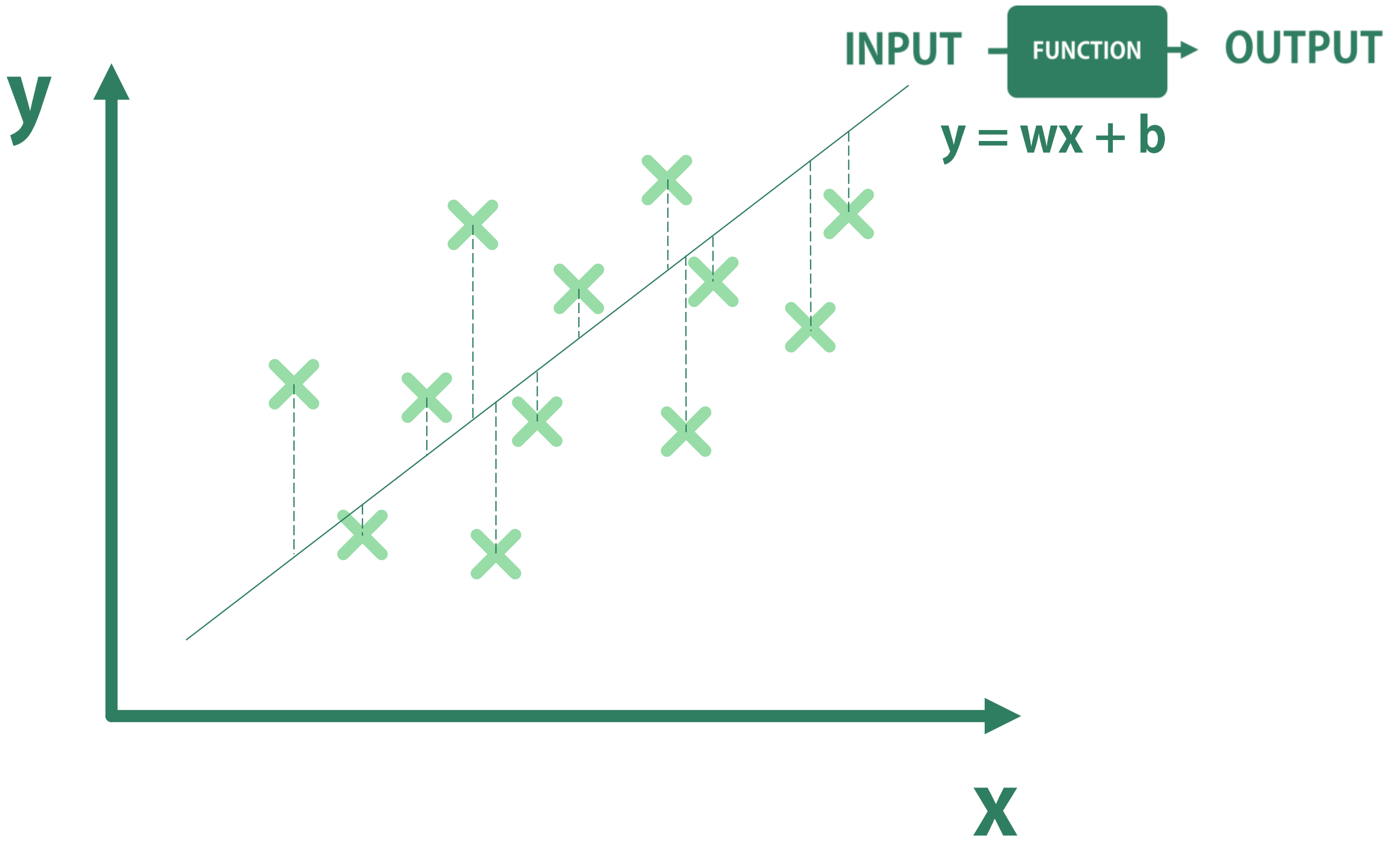
MACHINE LEARNING
PREDICTION

INPUT → FUNCTION → OUTPUT

GIVEN INPUT VARIABLES

PREDICT CLASS/ AMOUNT

LINEAR REGRESSION
BACK TO BASICS

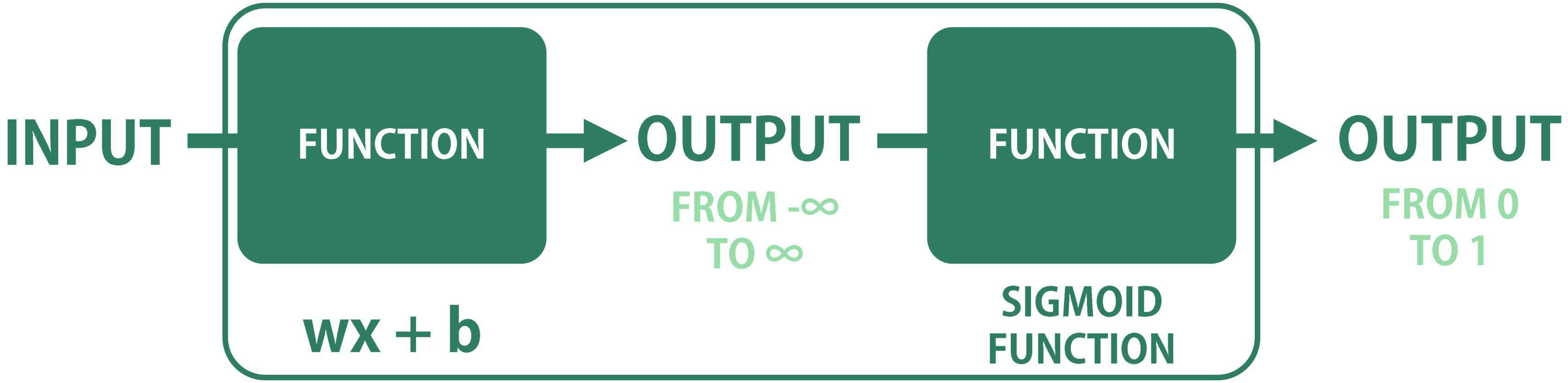


OPTIMIZE (FIND w AND b)
MINIMIZING MEAN SQUARED ERROR

LOGISTIC REGRESSION

USE LINEAR REGRESSION TO PREDICT CLASS

OUTPUT FROM LINEAR REGRESSION IS FROM -∞ TO ∞



OPTIMIZE (FIND w AND b)

MINIMIZING CROSS ENTROPY

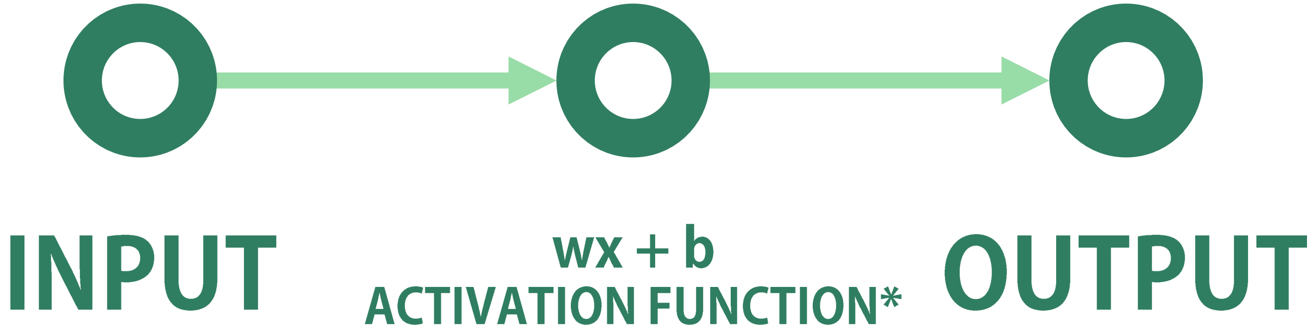
2

INTRODUCTION TO DEEP LEARNING

NEURAL NETWORKS

THE MOST BASIC NEURAL NETWORK

LINEAR/LOGISTIC REGRESSION

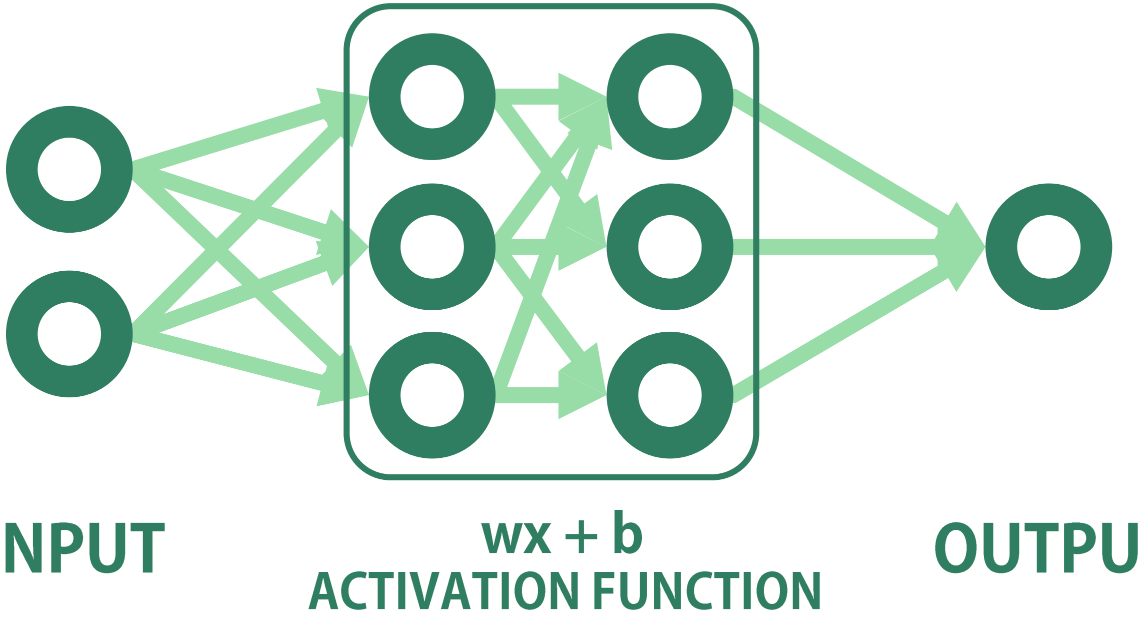


* ACTIVATION FUNCTION IS USED TO INTRODUCE NON-LINEARITY

SIGMOID FUNCTION, RELU, LEAKY RELU

NEURAL NETWORKS

COMPLEX NEURAL NETWORKS CAN BE BUILT WITH THE SAME IDEA



OPTIMIZE (FIND w AND b AT EACH NODE AND LAYER)
MINIMIZING MEAN SQUARED ERROR/CROSS ENTROPY

SOLUTION OVERVIEW

GENERATIVE ADVERSARIAL NETWORKS AND CONVOLUTIONAL NEURAL NETWORKS ARE USED IN THE CURRENT TASK

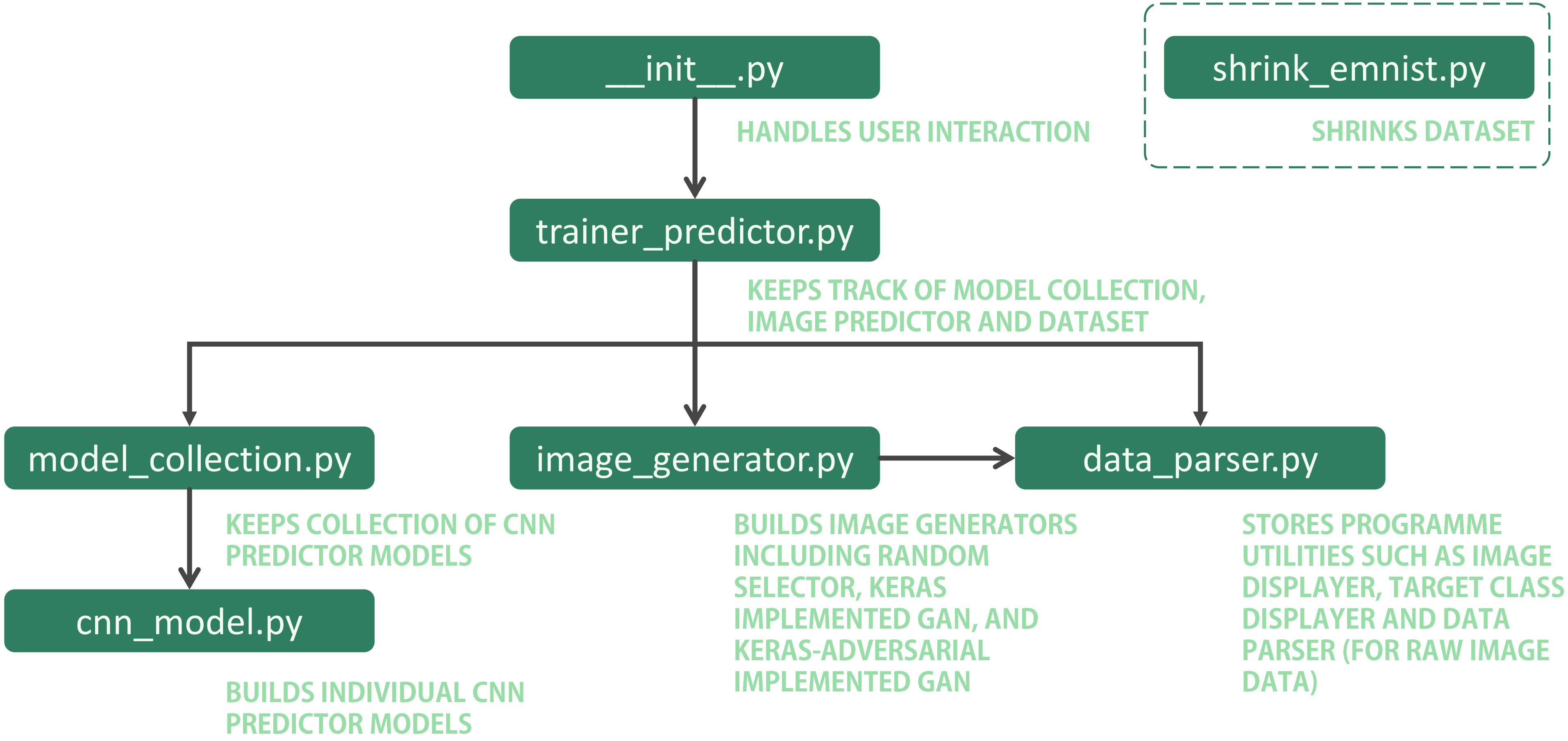
IMAGE GENERATOR > CHARACTER IMAGE > IMAGE PREDICTOR

- GENERATIVE ADVERSARIAL NETWORK (GAN)
    - KERAS IMPLEMENTATION
    - KERAS-ADVERSARIAL IMPLEMENTATION
- RANDOM SELECTION FROM SAMPLE

- CONVOLUTIONAL NEURAL NETWORKS (CNN)
    - TESTED 11 HYPERPARAMETER SETTINGS

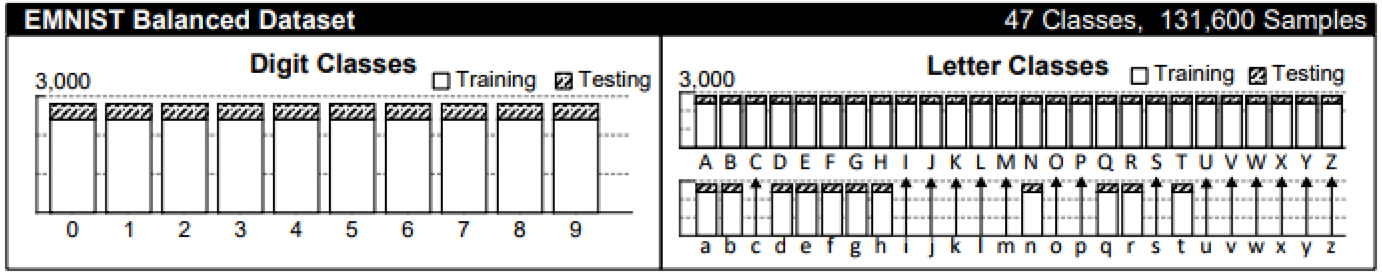
SOLUTION OVERVIEW

THE SOLUTION CONSISTS MAINLY OF 6 PYTHON FILES, WITH AN ADDITIONAL FILE TO SHRINK THE DATASET

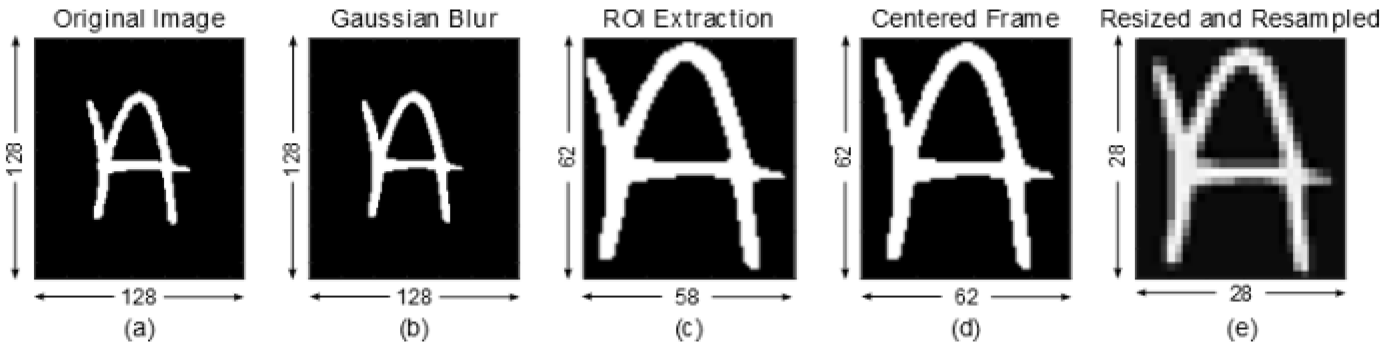


MODEL SELECTION PROCESS
DATASET

THE EMNIST BALANCED DATASET IS USED FOR THE CURRENT TASK
47 CLASSES – 10 DIGITS AND 37 CHARACTER CLASSES
15 LOWER CASE CHARACTER CLASSES MERGED WITH UPPER CASE
2,800 SAMPLES IN EACH CLASS
PRE-CLASSIFIED TRAINING AND TESTING SETS



EACH IMAGE IS 28 x 28 SIZE



3

SOLUTION

ALAN WONG

MODEL SELECTION PROCESS

DATA SPLITTING

DATA IS SPLIT INTO TRAINING, VALIDATION AND TESTING SETS

ALL SAMPLES (EXCLUDING DIGIT CLASSES)

TRAINING	TESTING
80%	20% MODEL ACCURACY

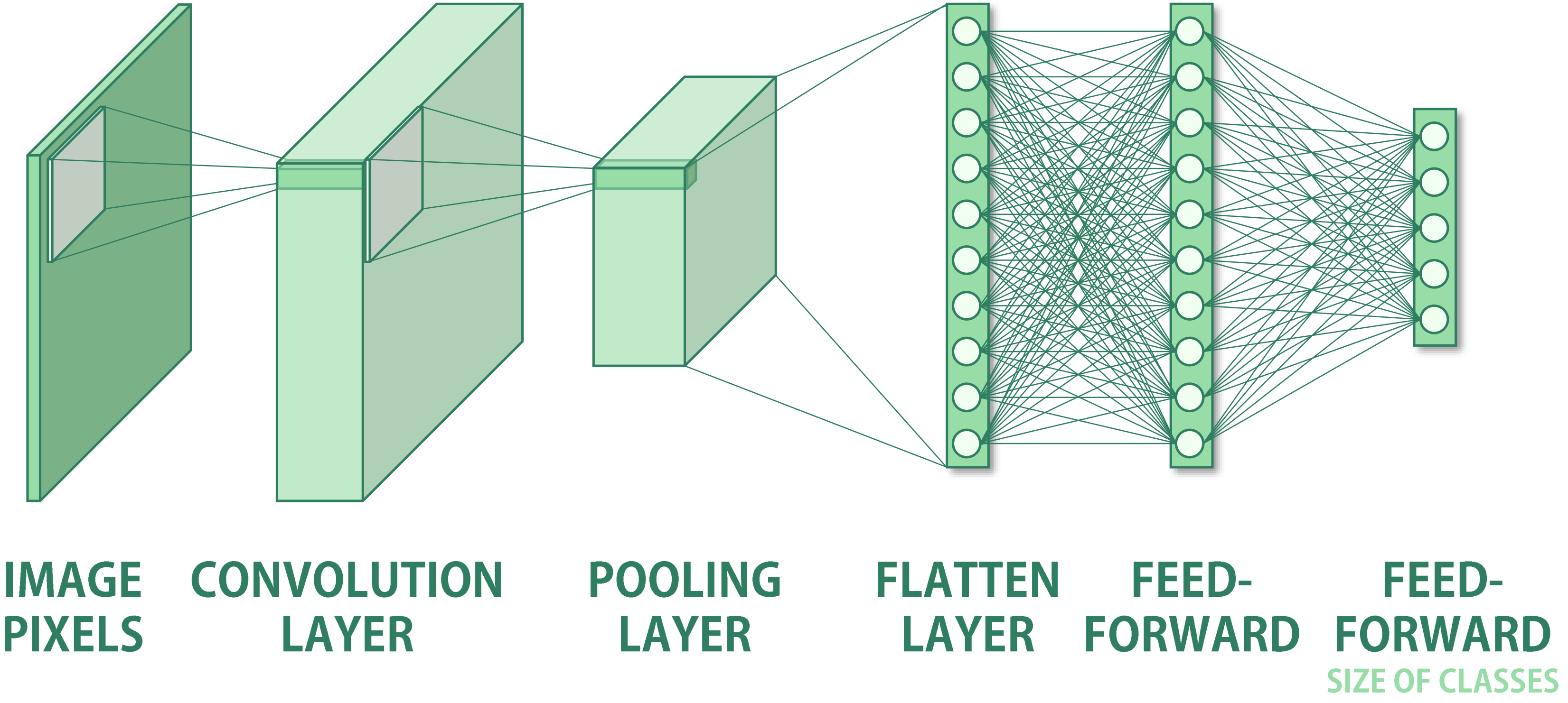
TRAINING	VALIDATION	TESTING	GAN
80% (64%) MODEL TRAINING	20% (16%) MODEL SELECTION	50% (10%) MODEL ACCURACY	50% (10%) GAN TRAINING

3

SOLUTION

CONVOLUTIONAL NEURAL NETWORKS (CNN)

CNN IS THE MOST COMMON CHOICE FOR IMAGE RECOGNITION



3A

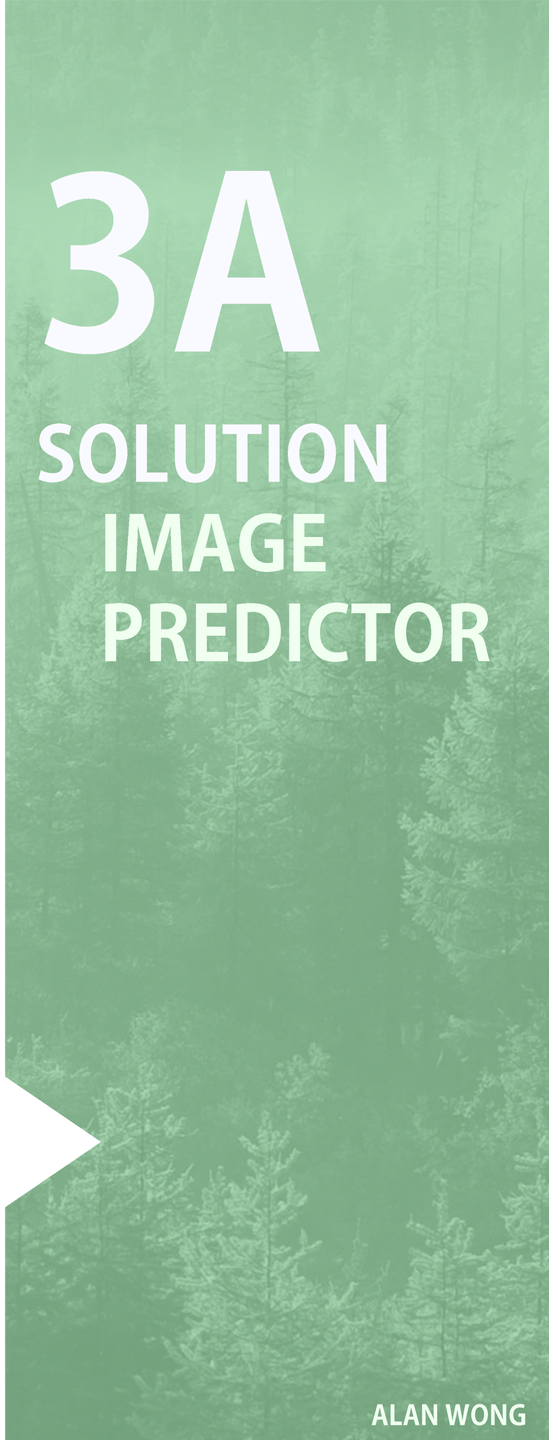
SOLUTION
IMAGE PREDICTOR

ALAN WONG

MODEL SELECTION PROCESS
MODELS TESTED

MODEL 7 GAVE THE BEST RESULTS ON VALIDATION SET
TEST SET ACCURACY IS 83.58%

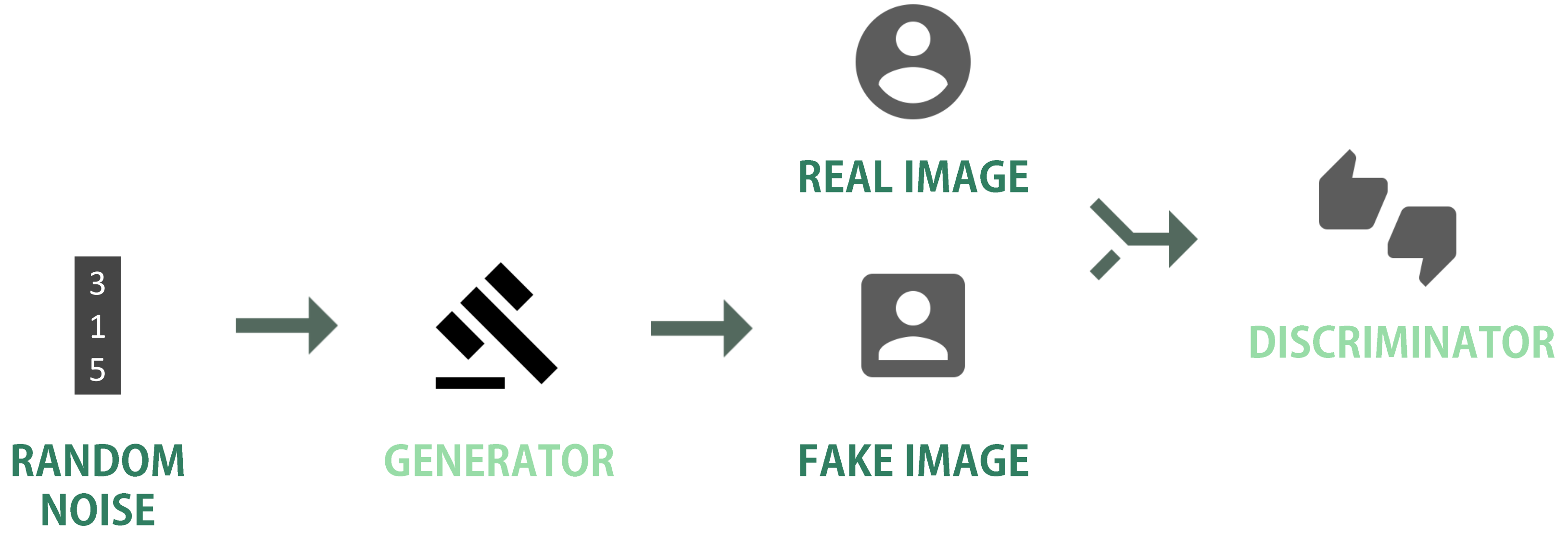
MODEL	CONVOLUTION LAYER 1 (FILTERS)	DROP OUT	CONVOLUTION LAYER 2 (FILTERS)	DROP OUT	FEED-FORWARD (NEURONS)	DROP OUT	FEED-FORWARD (NEURONS)	DROP OUT	RESULTS (VALID)
0	32	25%			64	25%			51.17%
1	64	25%			64	25%			50.78%
2	64	25%	32	25%	64	25%			39.42%
3	64	25%	64	25%	64	25%			41.52%
4	32	25%	32	25%	64	25%			40.50%
5	32	75%	32	50%	64	25%			53.79%
6	32	75%	32	75%	64	50%			77.15%
7	32	75%	32	75%	128	50%	64	50%	83.62%
8	64	75%	32	75%	128	50%	64	50%	76.51%
9	64	75%	64	75%	128	50%	64	50%	71.24%
10	64	75%	64	75%	256	50%	128	50%	57.59%



GENERATIVE ADVERSARIAL NETWORKS (GAN)

TWO NEURAL NETWORKS COMPETING WITH EACH OTHER

AT CONVERGE, DISCRIMINATOR UNABLE TO DISTINGUISH FAKE FROM REAL
GENERATOR CREATES REALISTIC IMAGES



CROSS ENTROPY
DISCRIMINATOR RECOGNIZES AS REAL

CROSS ENTROPY
DISCRIMINATOR DISTINGUISHES REAL AND FAKE IMAGES

3B

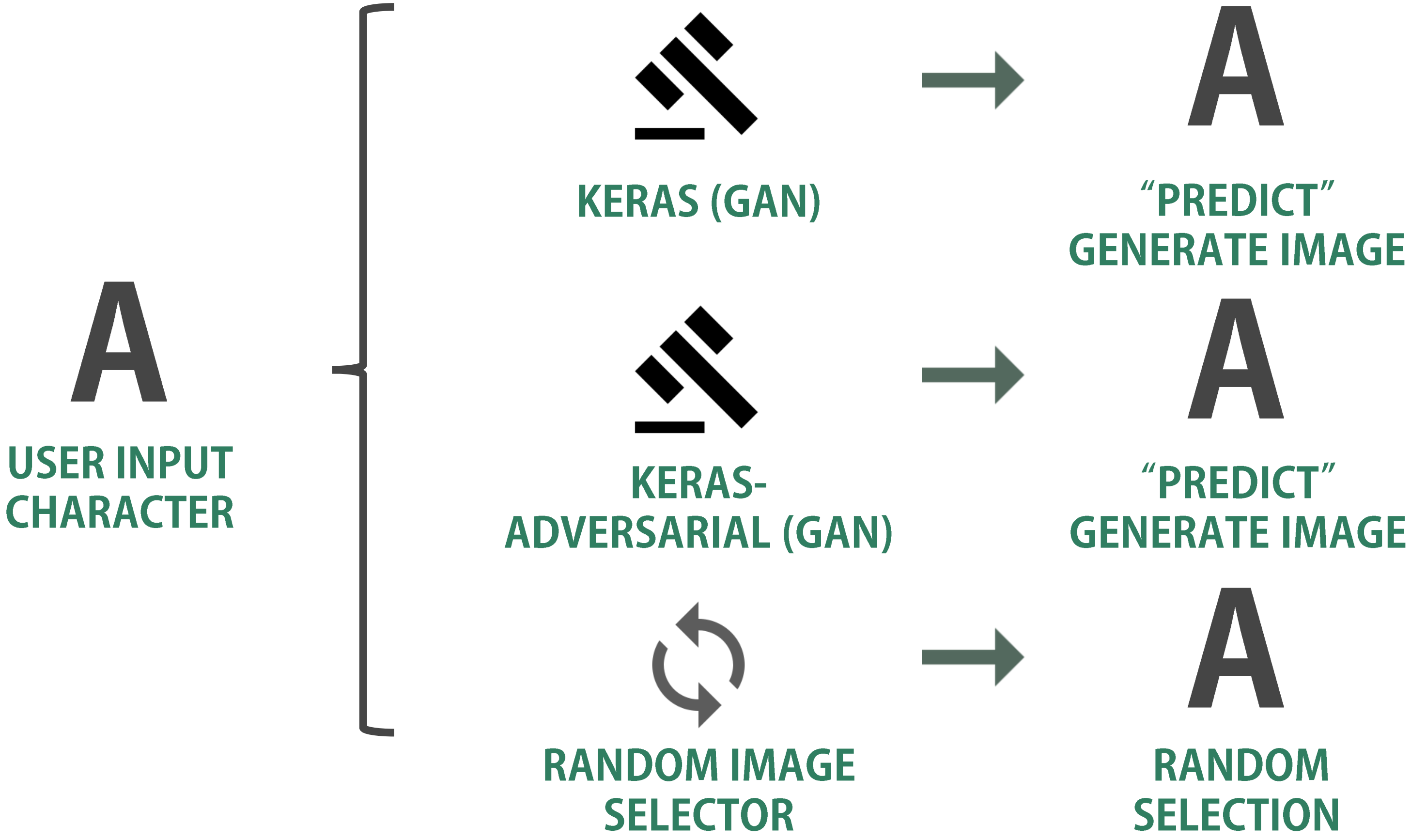
SOLUTION

IMAGE GENERATOR

ALAN WONG

IMAGE GENERATOR PROCESS

THE IMAGE GENERATOR ALLOWS USERS TO SELECT FROM ONE OF THREE IMPLEMENTATIONS



3B

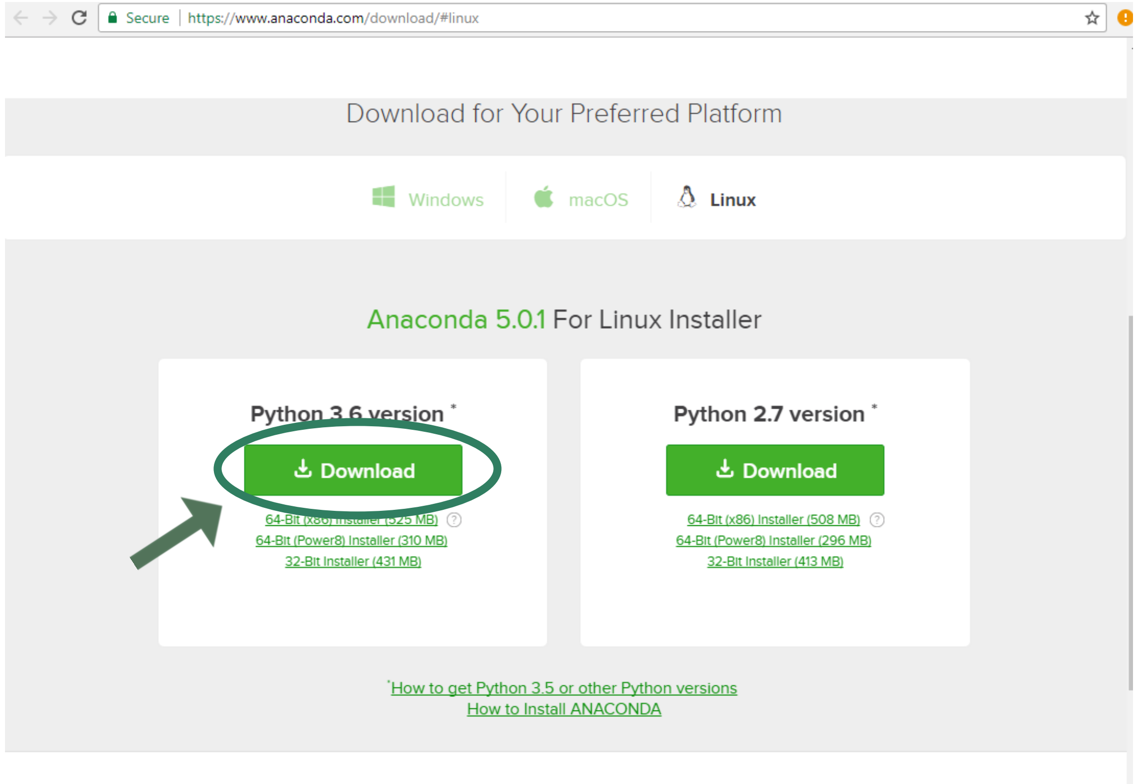
SOLUTION

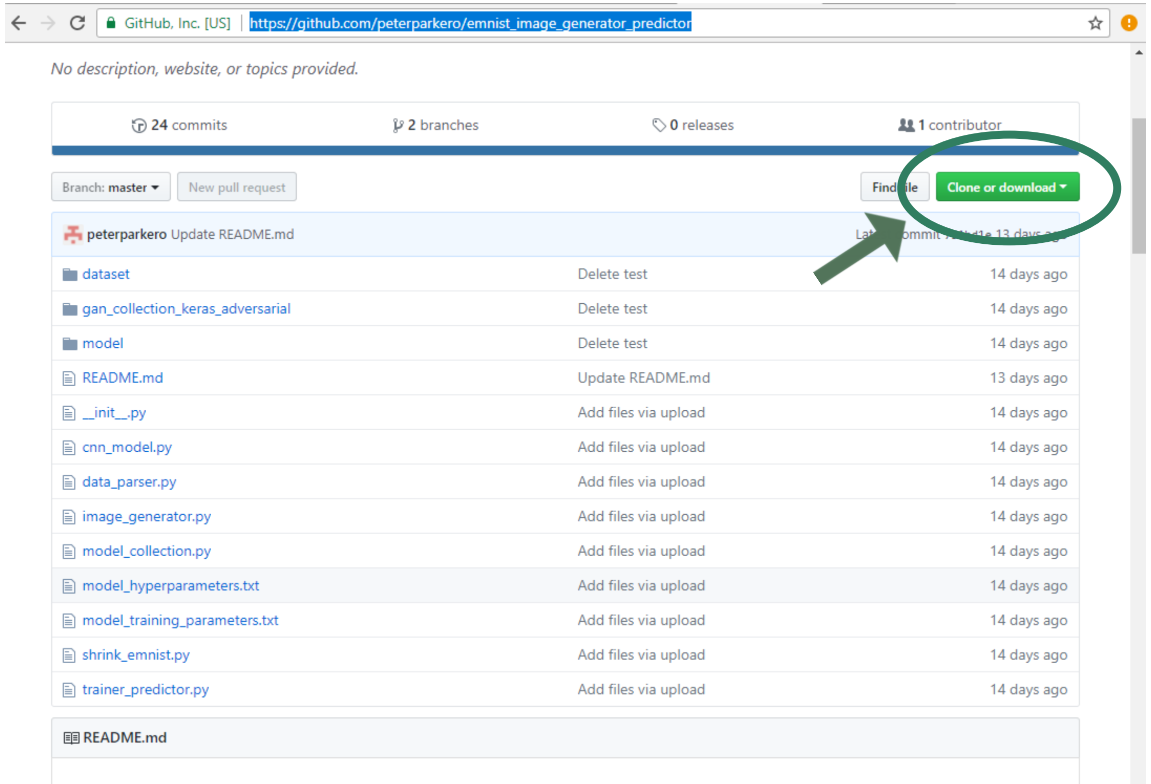
IMAGE GENERATOR

ALAN WONG

4 DEMONSTRATION

STEP 1 – INSTALL PYTHON (ANACONDA) AND REQUIRED PACKAGES
DOWNLOAD AND EXTRACT FILES FROM GITHUB





STEP 2 – OPEN TERMINAL/COMMAND PROMPT
GO TO FILE DIRECTORY AND EXECUTE

python __init__.py ./dataset/emnist-balanced-small.mat – m ./model

emnist_image_generator_predictor-master — -bash — 136×27

```
(venv) cplg104-mac17:emnist_image_generator_predictor-master cwong$ python __init__.py ./dataset/emnist-balanced-small.mat -m ./model
```

STEP 3 – REACH MAIN MENU UPON COMPLETION OF LOADING REQUIRED PACKAGES AND MODELS

WARNINGS FROM TENSORFLOW CAN BE IGNORED

emnist_image_generator_predictor-master — Python __init__.py ./dataset/emnist-balanced-small.mat -m ./model — 136×27

```
(venv) cplg104-mac17:emnist_image_generator_predictor-master cwong$ python __init__.py ./dataset/emnist-balanced-small.mat -m ./model
Using TensorFlow backend.
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/importlib/_bootstrap.py:205: RuntimeWarning: compiletime version 3.5 of
module 'tensorflow.python.framework.fast_tensor_util' does not match runtime version 3.6
  return f(*args, **kwds)
2017-11-17 20:19:43.124519: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow bin
ary was not compiled to use: SSE4.1 SSE4.2 AVX

Model collection file loaded.


Would you like to Train a Model, Generate and Predict a Image, Save Model Collection, Load Model Collection, Display Results, or Exit
[Train/Predict/Save/Load/Results/Exit]
```

TRAIN – TYPE "train" TO TRAIN NEW MODELS BASED ON SPECIFIED HYPERPARAMETER SETTINGS

ENTER HYPERPARAMETER AND TRAINING PARAMETER FILE PATHS

```
Would you like to Train a Model, Generate and Predict a Image, Save Model Collection, Load Model Collection, Display Results, or Exit
[Train/Predict/Save/Load/Results/Exit]
Train

Please input the hyperparameter_layer_filepath.
./model_hyperparameters.txt

Please input the training_parameters_filepath.
./model_training_parameters.txt

Please input whether you like to retrain the models if the same hyperparameter settings are given. [Y/N]
Y
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 14, 14, 32)        128
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                401472
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 47)                3055
=================================================================
Total params: 404,975
Trainable params: 404,911
Non-trainable params: 64
_________________________________________________________________
Epoch 1/10
2560/2560 [==============================] - 6s 2ms/step - loss: 3.4509 - acc: 0.1465
Epoch 2/10
2560/2560 [==============================] - 3s 1ms/step - loss: 2.4600 - acc: 0.3512
```

4 DEMONSTRATION

PREDICT – TYPE "predict" TO GENERATE AND PREDICT A CHARACTER IMAGE

ENTER "basic" FOR RANDOM IMAGE SELECTOR

```
Would you like to Train a Model, Generate and Predict a Image, Save Model Collection, Load Model Collection, Display Results, or Exit
[Train/Predict/Save/Load/Results/Exit]
Predict

Which image generator should be used? [basic/keras/keras_adversarial]
basic

What character would you like to generate image on and make a prediction?
Note that not all characters are available for prediction.
Unavailable characters are: c, i, j, k, l, m, o, p, s, u, v, w, x, y and z.
a
Preview of selected image:



                        ************

                  ************************

               ***************************

            *********************************

            ******************      ************

            ***************         ************

            ***************            ************

            ***************            ************

            ***************         ******************

            *********************************************

            ***************************************************

               ***************************************************

                  ************************      *********************

                                                      ******************

                                                            ************


The performance of this model:
Validation Score - 83.62%
Test Set Score - 83.58%

This character image is predicted to be:
a
```

PREDICT – TYPE “predict” TO GENERATE AND PREDICT A CHARACTER IMAGE

ENTER “keras” FOR GAN IMPLEMENTED USING KERAS

```
Would you like to Train a Model, Generate and Predict a Image, Save Model Collection, Load Model Collection, Display Results, or Exit
[Train/Predict/Save/Load/Results/Exit]
Predict

Which image generator should be used? [basic/keras/keras_adversarial]
keras

What character would you like to generate image on and make a prediction?
Note that not all characters are available for prediction.
Unavailable characters are: c, i, j, k, l, m, o, p, s, u, v, w, x, y and z.
a
Preview of selected image:
                                       ***************

                                    *********************

                                 ************************

                              ***************************

                           ******************************

                        *********************************

                        ************         ************

                  ************               ************

                  ************               ************

               ************               ***************

               *********                  ************

            ************               ***************

            ************               ***************

         ************               ******************

         ************         ************************

         ************      ***************************

         ***************************************************************

         ******************************      ************************

            ************************         ************************

               ***************                     *********

The performance of this model:
Validation Score - 83.62%
Test Set Score - 83.58%

This character image is predicted to be:
a
```

4 DEMONSTRATION

PREDICT - TYPE “predict” TO GENERATE AND PREDICT A CHARACTER IMAGE

ENTER “keras_adversarial” FOR GAN IMPLEMENTED USING KERAS-ADVERSARIAL

```
Would you like to Train a Model, Generate and Predict a Image, Save Model Collection, Load Model Collection, Display Results, or Exit
[Train/Predict/Save/Load/Results/Exit]
Predict

Which image generator should be used? [basic/keras/keras_adversarial]
keras_adversarial

What character would you like to generate image on and make a prediction?
Note that not all characters are available for prediction.
Unavailable characters are: c, i, j, k, l, m, o, p, s, u, v, w, x, y and z.
a
Preview of selected image:

                                            ************

                                         ******************

                                      ************************

                                   ******************************

                                *********************************

                             ******************      ************

                          *********************      *********

                       *********************      ***************

                       ******************      ******************

                    ***************         *********************

                    ***************         *********************

                    ***************      ************************

                    *********************************************

                    *********************************************

                    *********************************************

                    *********************************************

                       ************************   ******************

                          ***************            ***************

                                                     ***************

                                                        ************

                                                        ************

                                                                 ***


The performance of this model:
Validation Score - 83.62%
Test Set Score - 83.58%

This character image is predicted to be:
a
```

SAVE – TYPE "save" TO SAVE THE CURRENT MODEL COLLECTION TO A FOLDER

```
Would you like to Train a Model, Generate and Predict a Image, Save Model Collection, Load Model Collection, Display Results, or Exit
[Train/Predict/Save/Load/Results/Exit]
Save

Please input the output folder path.
./model

Files saved to ./model.
```

LOAD – TYPE "load" TO LOAD A PREVIOUSLY SAVED MODEL COLLECTION FROM A FOLDER

LOADING THE PROGRAM WITHOUT "-m ./model" AND LOADING THE MODELS AFTER REACHING THE MENU PAGE IS EQUIVALENT TO RUNNING "python __init__.py ./dataset/emnist-balanced-small.mat – m ./model"

```
Would you like to Train a Model, Generate and Predict a Image, Save Model Collection, Load Model Collection, Display Results, or Exit
[Train/Predict/Save/Load/Results/Exit]
Load

Please input the folder path to load the model collection from.
./model

Model collection file loaded.
```

RESULTS – TYPE “results” TO DISPLAY THE VALIDATION SCORES OF ALL TRAINED MODELS FROM THE MODEL COLLECTION

emnist_image_generator_predictor-master — Python __init__.py ./dataset/emnist-balanced-small.mat -m ./model — 136×51

```
Would you like to Train a Model, Generate and Predict a Image, Save Model Collection, Load Model Collection, Display Results, or Exit
[Train/Predict/Save/Load/Results/Exit]
Results

Display hyperparameters? [Y/N]
Y

Model 0
('conv2d', 32, 3, (1, 1), 'same', 'relu')
('maxpooling', (2, 2))
('dropout', 0.25)
batchnormalization
flatten
('dense', 64, 'relu')
('dropout', 0.25)
    Validation Score - 51.17%


Model 1
('conv2d', 64, 3, (1, 1), 'same', 'relu')
('maxpooling', (2, 2))
('dropout', 0.25)
batchnormalization
flatten
('dense', 64, 'relu')
('dropout', 0.25)
    Validation Score - 50.78%


Model 2
('conv2d', 64, 3, (1, 1), 'same', 'relu')
('maxpooling', (2, 2))
('dropout', 0.25)
batchnormalization
('conv2d', 32, 3, (1, 1), 'same', 'relu')
```

EXIT – TYPE “exit” TO EXIT THE PROGRAM

```
Would you like to Train a Model, Generate and Predict a Image, Save Model Collection, Load Model Collection, Display Results, or Exit
[Train/Predict/Save/Load/Results/Exit]
Exit
(venv) cplg104-mac17:emnist_image_generator_predictor-master cwong$
```

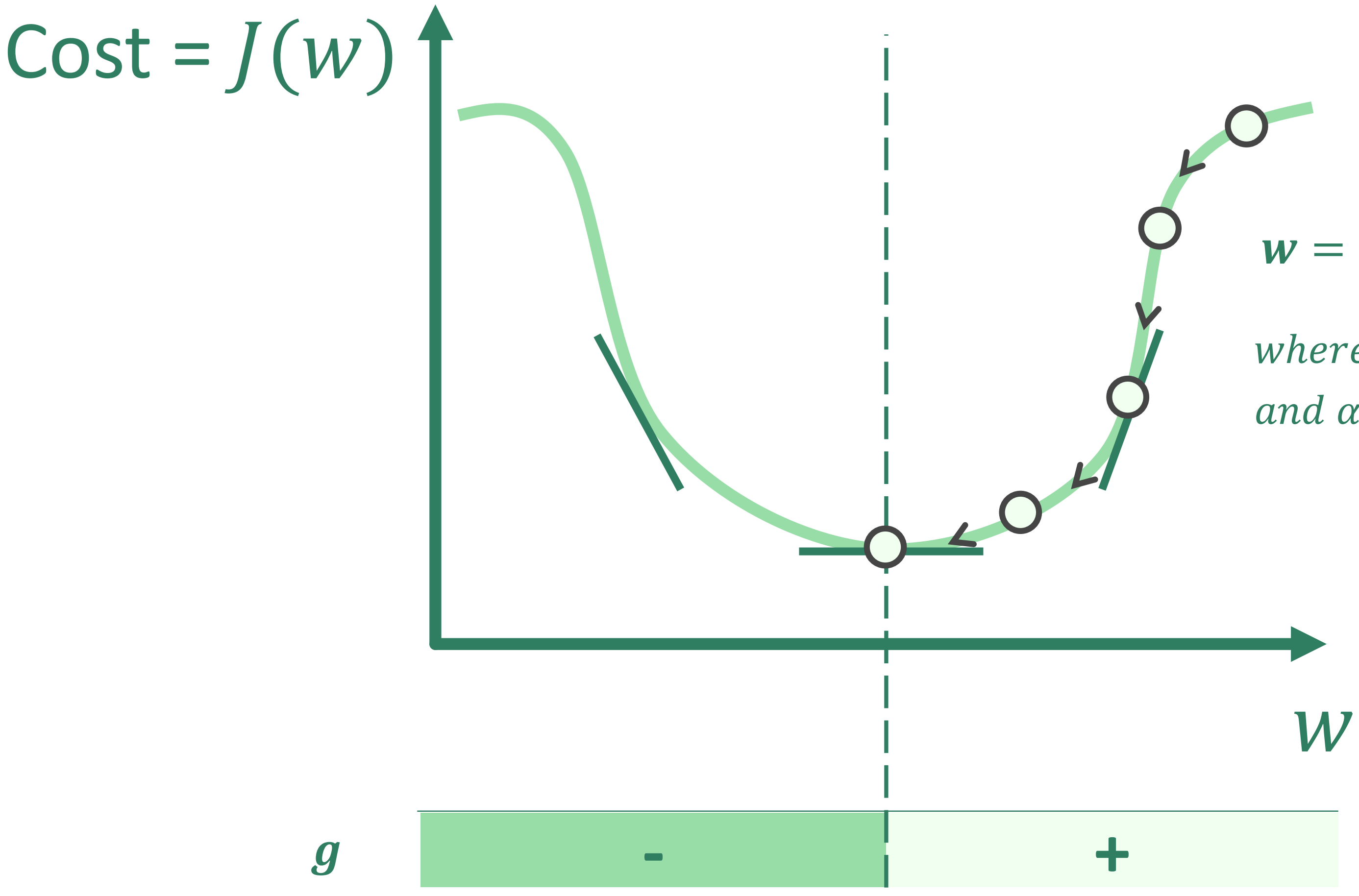
APPENDIX

- GRADIENT DESCENT
- ADAM
- BATCH NORMALIZATION
- BACKPROPAGATION
- DROPOUT
- FORMULA

5

APPENDIX

GRADIENT DESCENT



$$\boldsymbol{w = w - \alpha g}$$

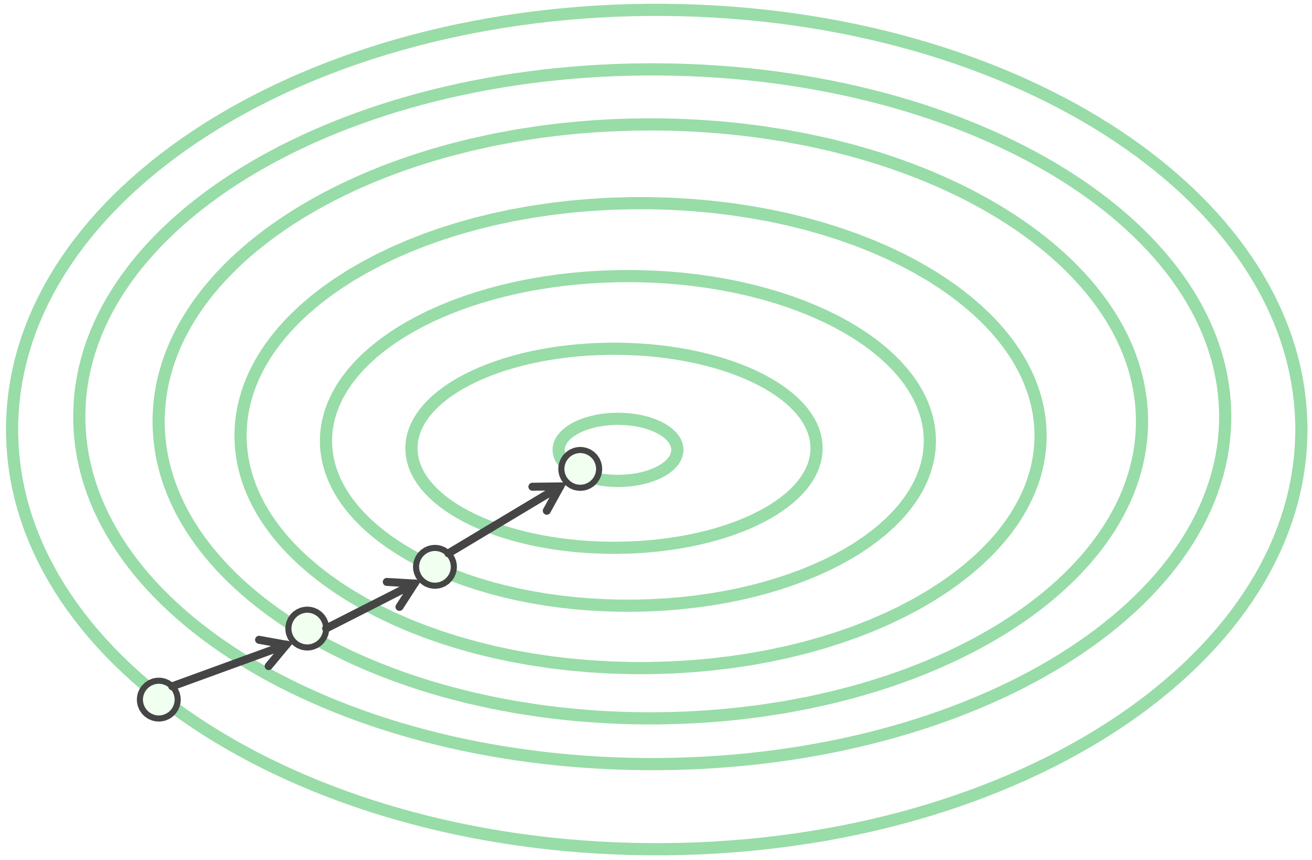
where $g = \frac{\partial}{\partial x} J(w)$

and α *is the learning rate*

5

APPENDIX

GRADIENT DESCENT



5

APPENDIX

OTHER OPTIMIZATION TECHNIQUES

MOMENTUM

INCLUDES A "MOMENTUM" FACTOR IN EACH UPDATE
LINEAR COMBINATION OF GRADIENT OF PREVIOUS UPDATE

$$\Delta w = -\alpha g + \mu \Delta w$$
$$w = w + \Delta w$$

ADAPTIVE GRADIENT (ADAGRAD)

INCREASES LEARNING RATE FOR MORE SPARSE PARAMETERS
WORKS WELL WITH **SPARSE GRADIENTS**

$$w_t = w_{t-1} - \alpha \frac{g_{t-1}}{\sqrt{\sum_{i=1}^{t} g_i{}^2}}$$

ROOT MEAN SQUARE PROPAGATION (RMSPROP)

EXPONENTIAL MOVING AVERAGE CONTROLLED BY γ
WORKS WELL IN **NON-STATIONARY SETTINGS** (E.G. TIME SERIES)

$$R_t = \gamma R_{t-1} + (1-\gamma) g_{t-1}{}^2$$
$$w_t = w_{t-1} - \alpha \frac{g_{t-1}}{\sqrt{R_t}}$$

5

APPENDIX

OTHER OPTIMIZATION TECHNIQUES

ADAPTIVE MOMENT ESTIMATION (ADAM)

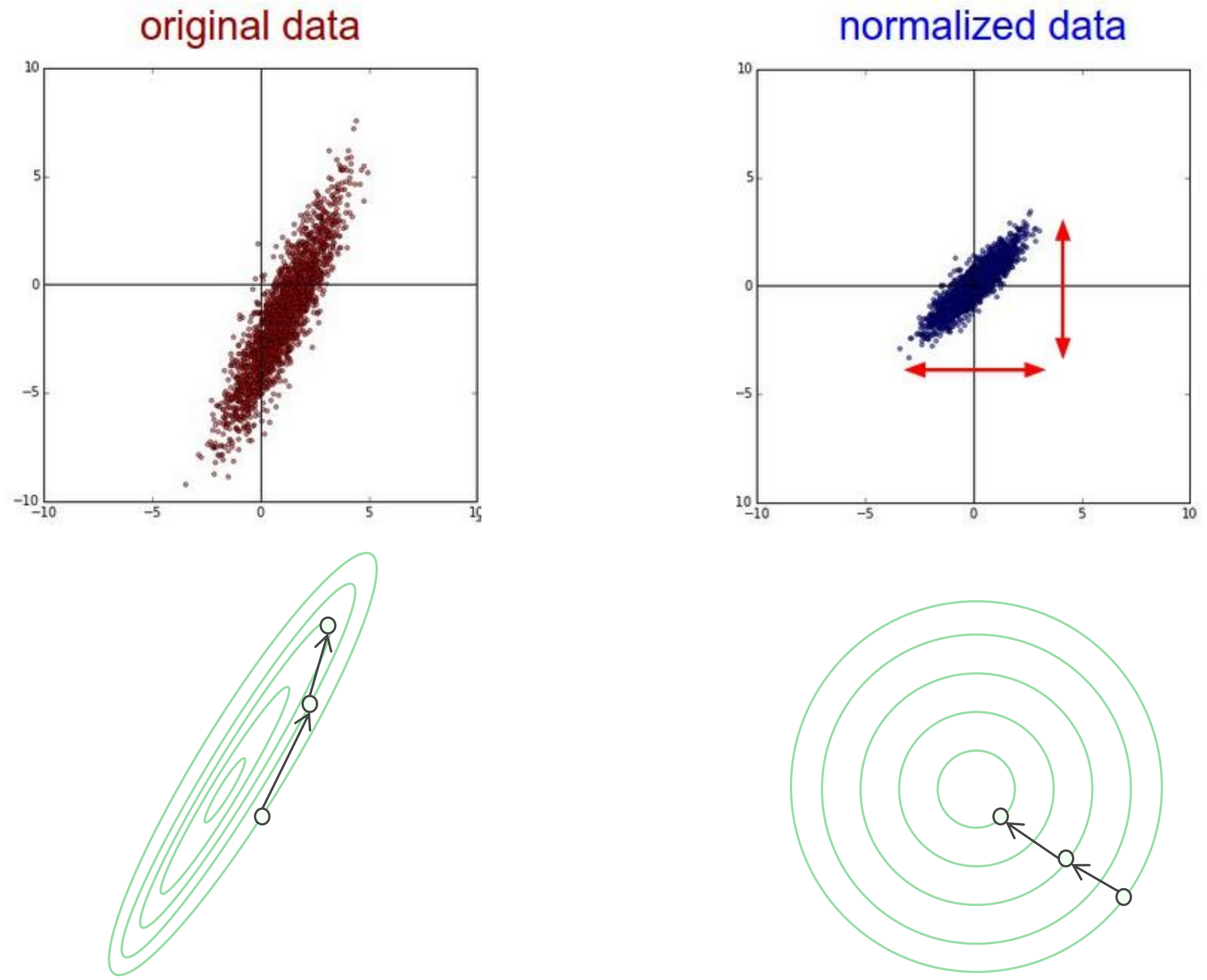
COMBINE THE ADVANTAGES OF ADAGRAD AND RMSPROP

$M_0 = 0, R_0 = 0$	INITIALIZE
$For\ t = 1\ to\ T$:	
$M_t = \beta_1 M_{t-1} + (1-\beta_1) g_{t-1}$	1ST MOMENT ESTIMATE
$R_t = \beta_2 R_{t-1} + (1-\beta_2)(g_{t-1})^2$	2ND MOMENT ESTIMATE
$\widehat{M_t} = \frac{M_t}{1-\beta_1{}^t}$	1ST MOMENT BIAS CORRECTION
$\widehat{R_t} = \frac{R_t}{1-\beta_2{}^t}$	2ND MOMENT BIAS CORRECTION
$W_t = W_{t-1} - \alpha \frac{\widehat{M_t}}{\sqrt{\widehat{R_t}}+\varepsilon}$	UPDATE PROPORTIONAL TO $\frac{average\ gradient}{\sqrt{average\ squared\ gradient}}$

5

APPENDIX

BATCH NORMALIZATION



5

APPENDIX

BACKPROPAGATION

x, w, b $\rightarrow$ $z = wx + b$ $\rightarrow$ $\hat{y} = \sigma(z)$ $\rightarrow$ $L(\hat{y}, y)$

CROSS ENTROPY

$\frac{dL}{dw}$ $\frac{dL}{db}$

x, w, b $\leftarrow$ $\frac{dL}{dz}$ $\leftarrow$ $\frac{dL}{d\hat{y}}$ $\leftarrow$ $L(\hat{y}, y)$

$$= \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dz}$$

APPLY CHAIN RULE

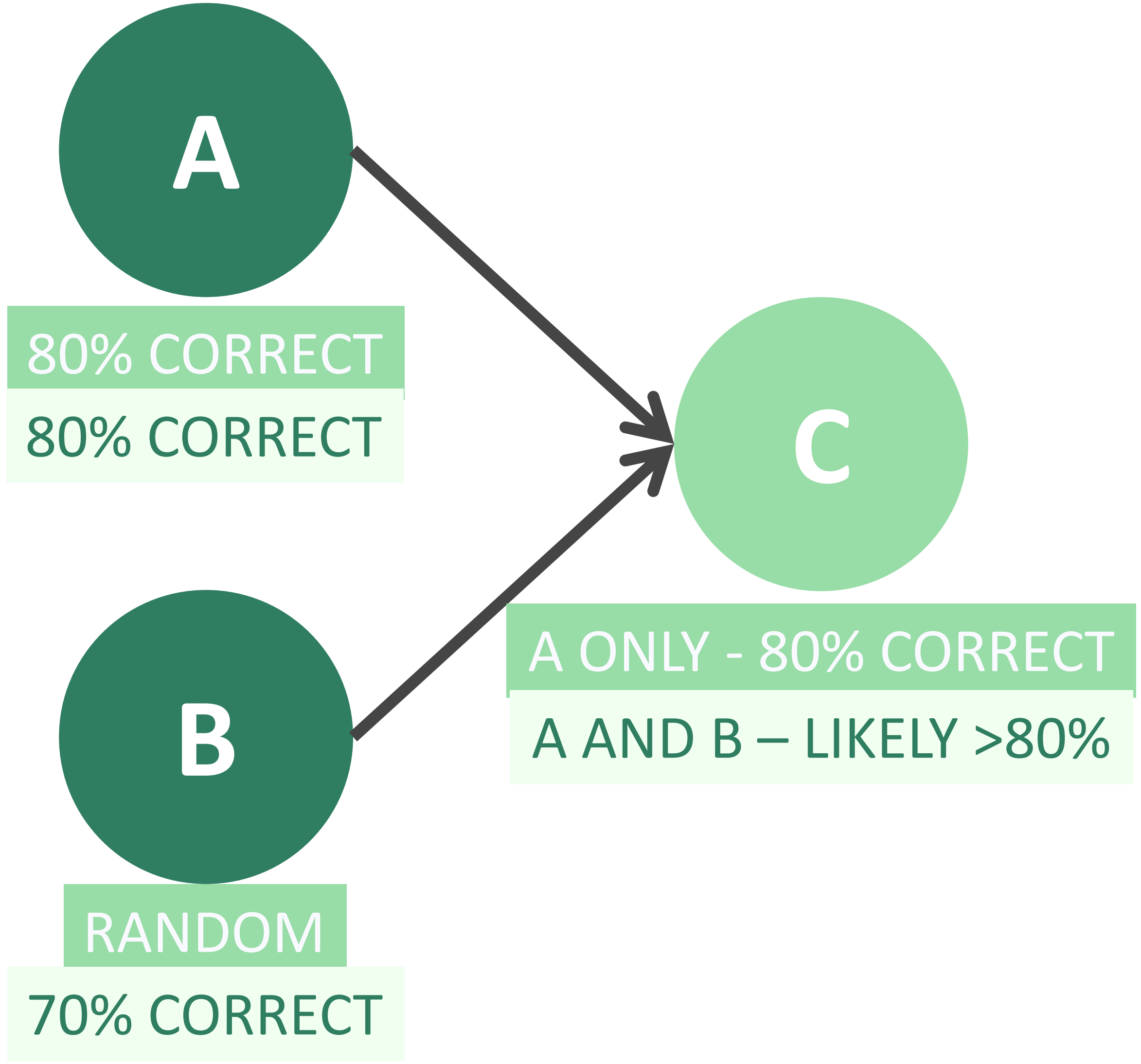
APPLY CHAIN RULE

* **CROSS ENTROPY:** $\boldsymbol{L(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})}$

5

APPENDIX

DROPOUT



5

APPENDIX

FORMULA

LINEAR REGRESSION

$$y = \sum_{i=1}^{n} w_i x_i + b$$

MEAN SQUARED ERROR

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{Y_i} - Y_i\right)^2$$

LOGISTIC REGRESSION

$$y = \frac{1}{1+e^{-\left(\sum_{i=1}^{n} w_i x_i + b\right)}}$$

CROSS ENTROPY

$$CE = \sum_{i=1}^{n} y^{(i)}\, log\left(\widehat{y^{(i)}}\right) + \left(1 - y^{(i)}\right) log\left(1 - \widehat{y^{(i)}}\right)$$