Report

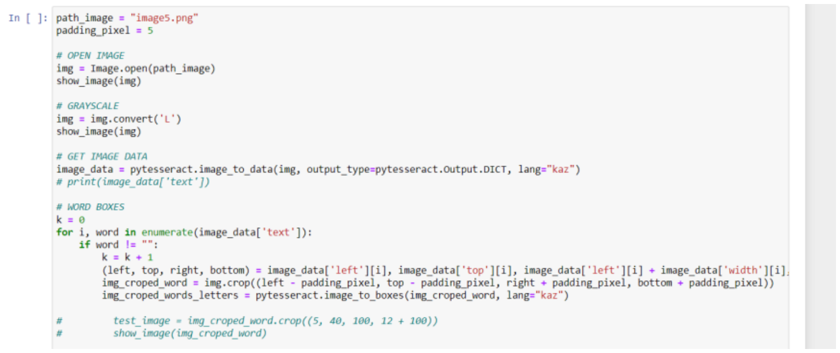
Optical Character Recognition (OCR) for Kazakh Language

Nazarbay Nursultan, Magzhan Omirtay

Git link: https://github.com/NazarbayND/ai_final_project

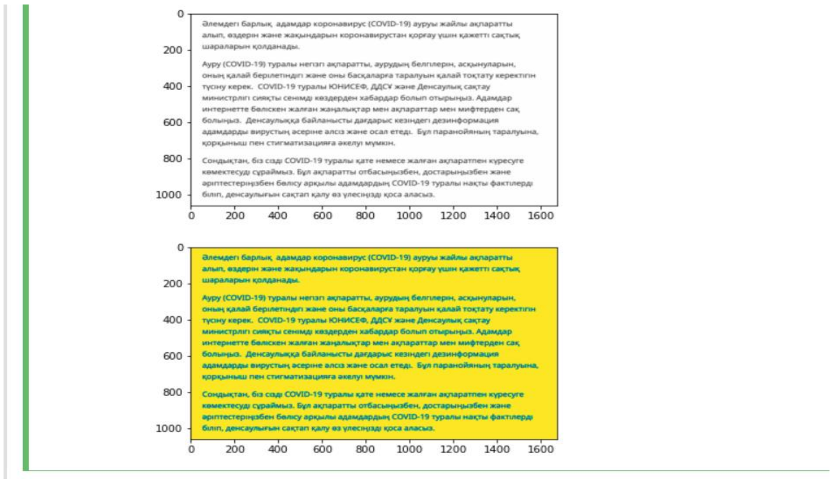
1. Image preprocessing

To solve this problem we have generated data by writing a python script for obtaining Kazakh letters and saved them as a given letter. The script receives a scanned or photographed Kazakh text using PIL and defines words using the pytesseract library. Initially, we tried to get the coordinates of the letters using the pytesseract itself, but it does not show the correct coordinates, although it recognizes the words correctly. For our script to work correctly, we decided to use cv2 library and processed our image with a threshold function. Then we found letter contours using the findContours function, but the contour does not understand the letters and selects all objects in the images.



```
In [ ]: path_image = "image5.png"
        padding_pixel = 5

        # OPEN IMAGE
        img = Image.open(path_image)
        show_image(img)

        # GRAYSCALE
        img = img.convert('L')
        show_image(img)

        # GET IMAGE DATA
        image_data = pytesseract.image_to_data(img, output_type=pytesseract.Output.DICT, lang="kaz")
        # print(image_data['text'])

        # WORD BOXES
        k = 0
        for i, word in enumerate(image_data['text']):
            if word != "":
                k = k + 1
                (left, top, right, bottom) = image_data['left'][i], image_data['top'][i], image_data['left'][i] + image_data['width'][i],
                img_croped_word = img.crop((left - padding_pixel, top - padding_pixel, right + padding_pixel, bottom + padding_pixel))
                img_croped_words_letters = pytesseract.image_to_boxes(img_croped_word, lang="kaz")

        #         test_image = img_croped_word.crop((5, 40, 100, 12 + 100))
        #         show_image(img_croped_word)
```

Picture 1 – code for converting image to grayscale



Әлемдегі барлық адамдар коронавирус (COVID-19) ауруы жайлы ақпаратты алып, өздерін және жақындарын коронавирустан қорғау үшін қажетті сақтық шараларын қолданады.

Ауру (COVID-19) туралы негізгі ақпаратты, аурудың белгілерін, асқынуларын, оның қалай берілетіндігін және оны басқаларға таралуын қалай тоқтату керектігін түсіну керек. COVID-19 туралы ЮНИСЕФ, ДДСҰ және Денсаулық сақтау министрлігі сияқты сенімді көздерден хабардар болып отырыңыз. Адамдар интернетте бөліскен жалған жаңалықтар мен ақпараттар мен мифтерден сақ болыңыз. Денсаулыққа байланысты дағдарыс кезіндегі дезинформация адамдарды вирустың әсеріне алсіз және осал етеді. Бұл паранойяның таралуына, қорқыныш пен стигматизацияға әкелуі мүмкін.

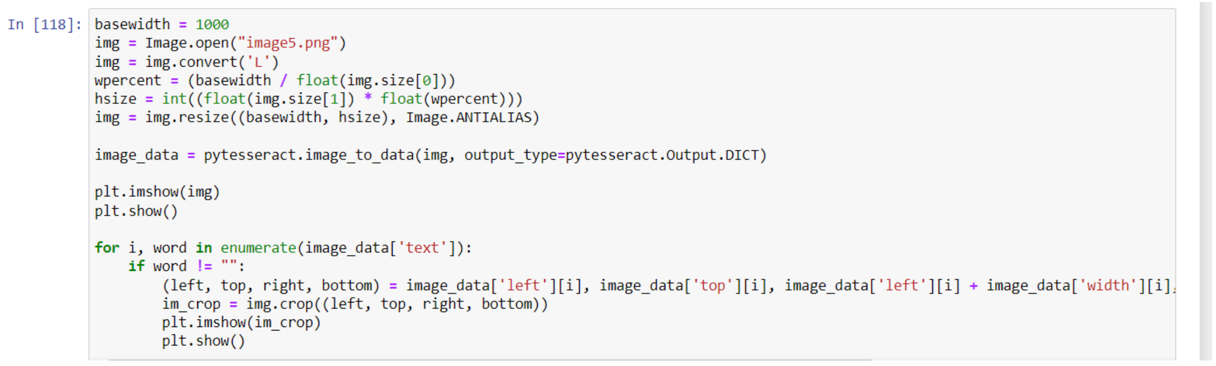
Сондықтан, біз сізді COVID-19 туралы қате немесе жалған ақпаратпен күресуге көмектесуді сұраймыз. Бұл ақпаратты отбасыңызбен, достарыңызбен және әріптестеріңізбен бөлісу арқылы адамдардың COVID-19 туралы нақты фактілерді біліп, денсаулығын сақтап қалу өз үлесіңізді қоса аласыз.

Әлемдегі барлық адамдар коронавирус (COVID-19) ауруы жайлы ақпаратты алып, өздерін және жақындарын коронавирустан қорғау үшін қажетті сақтық шараларын қолданады.

Ауру (COVID-19) туралы негізгі ақпаратты, аурудың белгілерін, асқынуларын, оның қалай берілетіндігін және оны басқаларға таралуын қалай тоқтату керектігін түсіну керек. COVID-19 туралы ЮНИСЕФ, ДДСҰ және Денсаулық сақтау министрлігі сияқты сенімді көздерден хабардар болып отырыңыз. Адамдар интернетте бөліскен жалған жаңалықтар мен ақпараттар мен мифтерден сақ болыңыз. Денсаулыққа байланысты дағдарыс кезіндегі дезинформация адамдарды вирустың әсеріне алсіз және осал етеді. Бұл паранойяның таралуына, қорқыныш пен стигматизацияға әкелуі мүмкін.

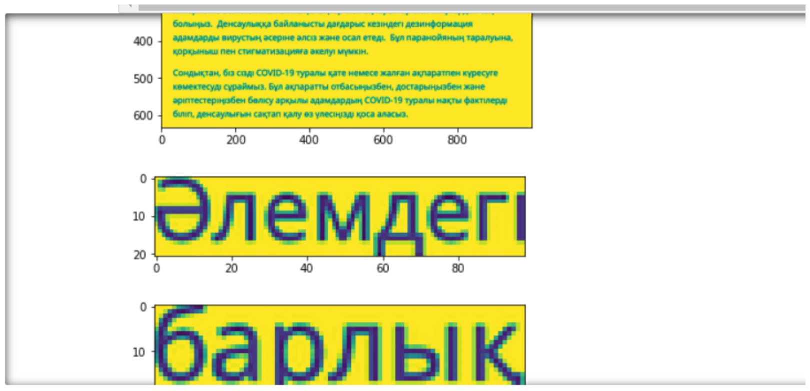
Сондықтан, біз сізді COVID-19 туралы қате немесе жалған ақпаратпен күресуге көмектесуді сұраймыз. Бұл ақпаратты отбасыңызбен, достарыңызбен және әріптестеріңізбен бөлісу арқылы адамдардың COVID-19 туралы нақты фактілерді біліп, денсаулығын сақтап қалу өз үлесіңізді қоса аласыз.

Picture 2 – result of converting



```
In [118]: basewidth = 1000
          img = Image.open("image5.png")
          img = img.convert('L')
          wpercent = (basewidth / float(img.size[0]))
          hsize = int((float(img.size[1]) * float(wpercent)))
          img = img.resize((basewidth, hsize), Image.ANTIALIAS)

          image_data = pytesseract.image_to_data(img, output_type=pytesseract.Output.DICT)

          plt.imshow(img)
          plt.show()

          for i, word in enumerate(image_data['text']):
              if word != "":
                  (left, top, right, bottom) = image_data['left'][i], image_data['top'][i], image_data['left'][i] + image_data['width'][i],
                  im_crop = img.crop((left, top, right, bottom))
                  plt.imshow(im_crop)
                  plt.show()
```

Picture 3 – code for splitting the words



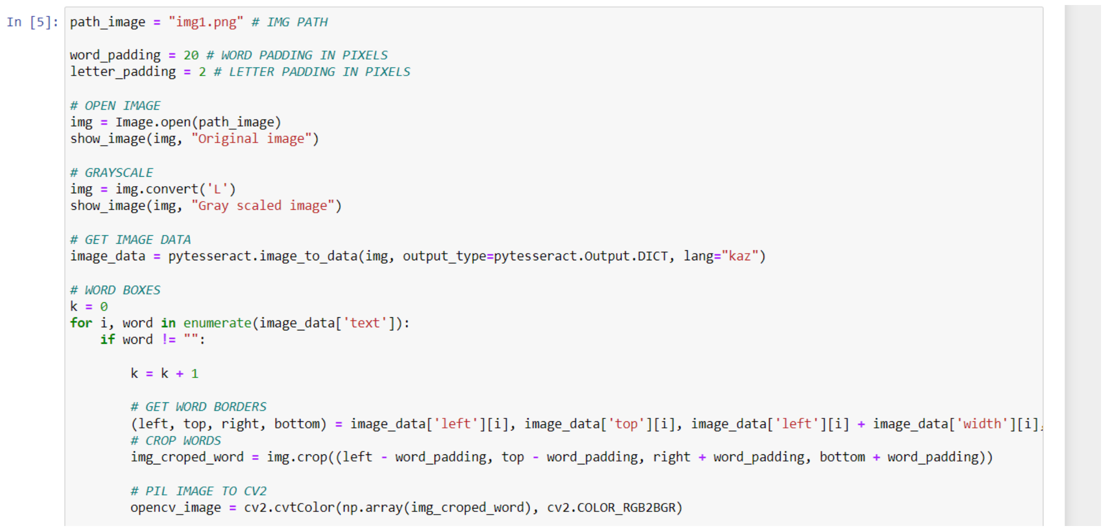
болыңыз. Денсаулыққа байланысты дағдарыс кезіндегі дезинформация адамдарды вирустың әсеріне әлсіз және осал етеді. Бұл паранойяның таралуына, қорқыныш пен стигматизацияға әкелуі мүмкін.

Сондықтан, біз сізді COVID-19 туралы қате немесе жалған ақпаратпен күресуге көмектесуді сұраймыз. Бұл ақпаратты отбасыңызбен, достарыңызбен және әріптестеріңізбен бөлісу арқылы адамдардың COVID-19 туралы нақты фактілерді біліп, денсаулығын сақтап қалу өз үлесіңізді қоса аласыз.

Әлемдегі

барлық

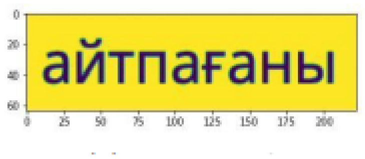
Picture 4 – result of splitting the words



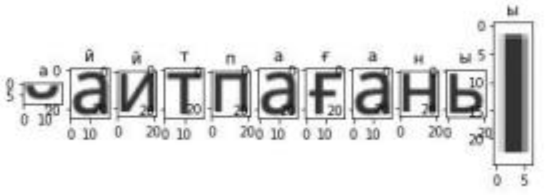
```
In [5]: path_image = "img1.png" # IMG PATH

        word_padding = 20 # WORD PADDING IN PIXELS
        letter_padding = 2 # LETTER PADDING IN PIXELS

        # OPEN IMAGE
        img = Image.open(path_image)
        show_image(img, "Original image")

        # GRAYSCALE
        img = img.convert('L')
        show_image(img, "Gray scaled image")

        # GET IMAGE DATA
        image_data = pytesseract.image_to_data(img, output_type=pytesseract.Output.DICT, lang="kaz")

        # WORD BOXES
        k = 0
        for i, word in enumerate(image_data['text']):
            if word != "":

                k = k + 1

                # GET WORD BORDERS
                (left, top, right, bottom) = image_data['left'][i], image_data['top'][i], image_data['left'][i] + image_data['width'][i],
                # CROP WORDS
                img_croped_word = img.crop((left - word_padding, top - word_padding, right + word_padding, bottom + word_padding))

                # PIL IMAGE TO CV2
                opencv_image = cv2.cvtColor(np.array(img_croped_word), cv2.COLOR_RGB2BGR)
```

Picture 5 – code for splitting words into letters

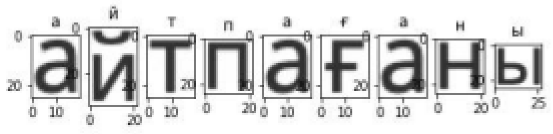
While processing our images, we ran into a problem with the letters "ы"and "й". Because our script split them into two parts and considered them to be different letters as in the Picture I (b). To solve this problem, we merged two adjacent images where these letters met. For this, additional functions were created: horizontal union for "ы"and vertical union for "й". Consequently script separates the words into letters for every 42 Kazakh letters. Further we saved them as a separate image as in the Picture 6. and this process occurred until we collect enough data for data augmentation. After this process, we increase the amount of data to improve model accuracy.



a) given word



b) initial separation with errors



c) result after script

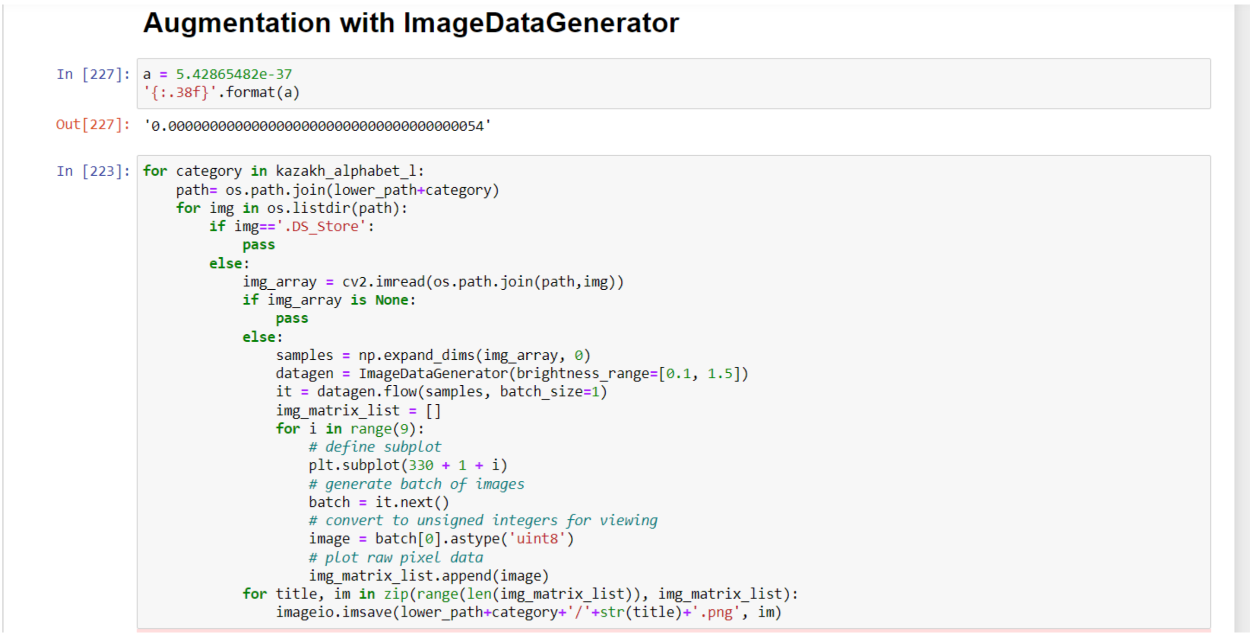
Picture 6 - Splitting words into letters

2. Image data generation

To get more data, we needed to make minor alterations to our existing dataset. Minor changes such as flips or translations or rotations. For this we have used data augmentation: a technique to increase the diversity of the training set by applying random (but realistic) transformations. Applying these small amounts of variations on the original image does not change its target class but only provides a new perspective of capturing the object in real life. It is quite often used for building deep learning models.

In our work we used several techniques of data augmentation such as Keras ImageDataGenerator, Albumentations Augmentations, ‘imgaug‘ Based Augmentations, TensorFlow-Based Augmentations.

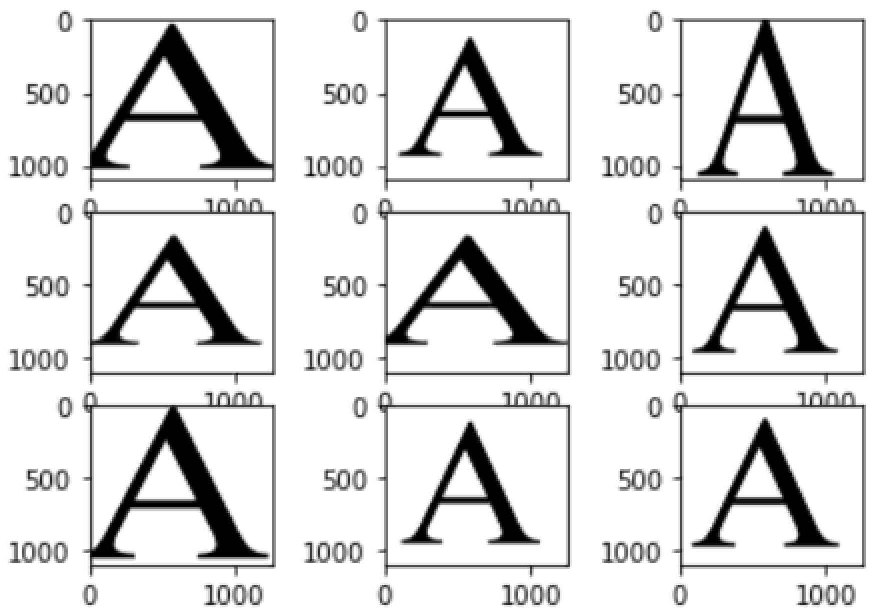
Keras ImageDataGenerator 1 class provides a quick and easy way to augment the images. It provides a host of different augmentation techniques like standardization, rotation, shifts, flips, brightness change, and many more. However, the main benefit of using the Keras ImageDataGenerator class is that it is designed to provide real-time data augmentation. Meaning it is generating augmented images on the fly while the model is still in the training stage.



Augmentation with ImageDataGenerator

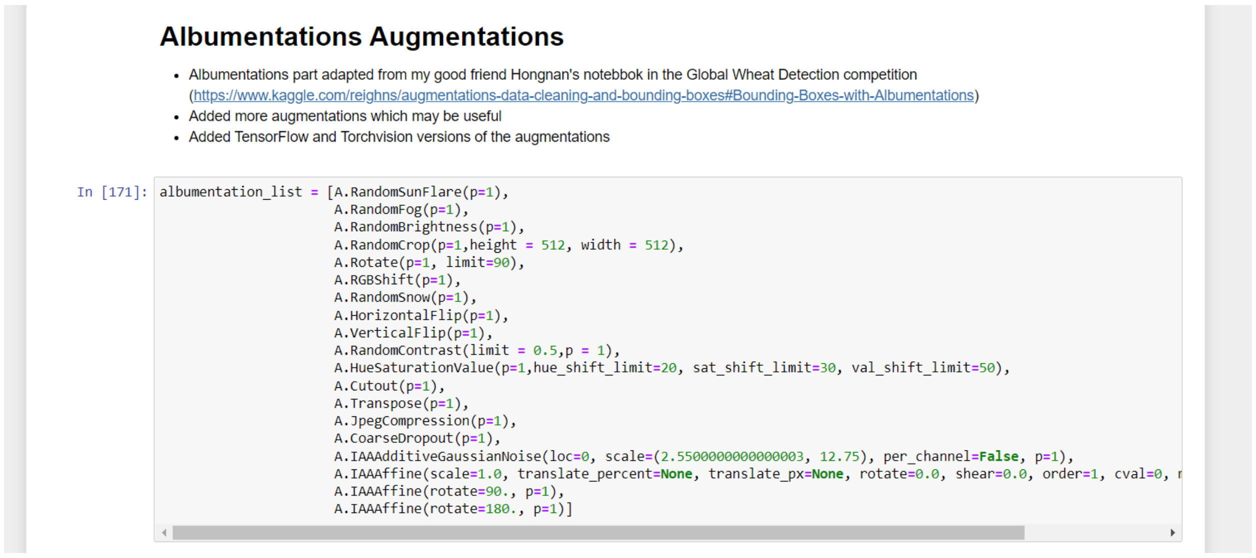
```
In [227]: a = 5.42865482e-37
          '{:.38f}'.format(a)

Out[227]: '0.00000000000000000000000000000000000054'

In [223]: for category in kazakh_alphabet_l:
              path= os.path.join(lower_path+category)
              for img in os.listdir(path):
                  if img=='.DS_Store':
                      pass
                  else:
                      img_array = cv2.imread(os.path.join(path,img))
                      if img_array is None:
                          pass
                      else:
                          samples = np.expand_dims(img_array, 0)
                          datagen = ImageDataGenerator(brightness_range=[0.1, 1.5])
                          it = datagen.flow(samples, batch_size=1)
                          img_matrix_list = []
                          for i in range(9):
                              # define subplot
                              plt.subplot(330 + 1 + i)
                              # generate batch of images
                              batch = it.next()
                              # convert to unsigned integers for viewing
                              image = batch[0].astype('uint8')
                              # plot raw pixel data
                              img_matrix_list.append(image)
                      for title, im in zip(range(len(img_matrix_list)), img_matrix_list):
                          imageio.imsave(lower_path+category+'/'+str(title)+'.png', im)
```

Picture 7 – code for ImageDataGenerator augmentation



Picture 8 - ImageDataGenerator results

Albumentations 2 is a Python library for fast and flexible image augmentations. Albumentations efficiently implements a rich variety of image transform operations that are optimized for performance, and does so while providing a concise, yet powerful image augmentation interface for different computer vision tasks, including object classification, segmentation, and detection.

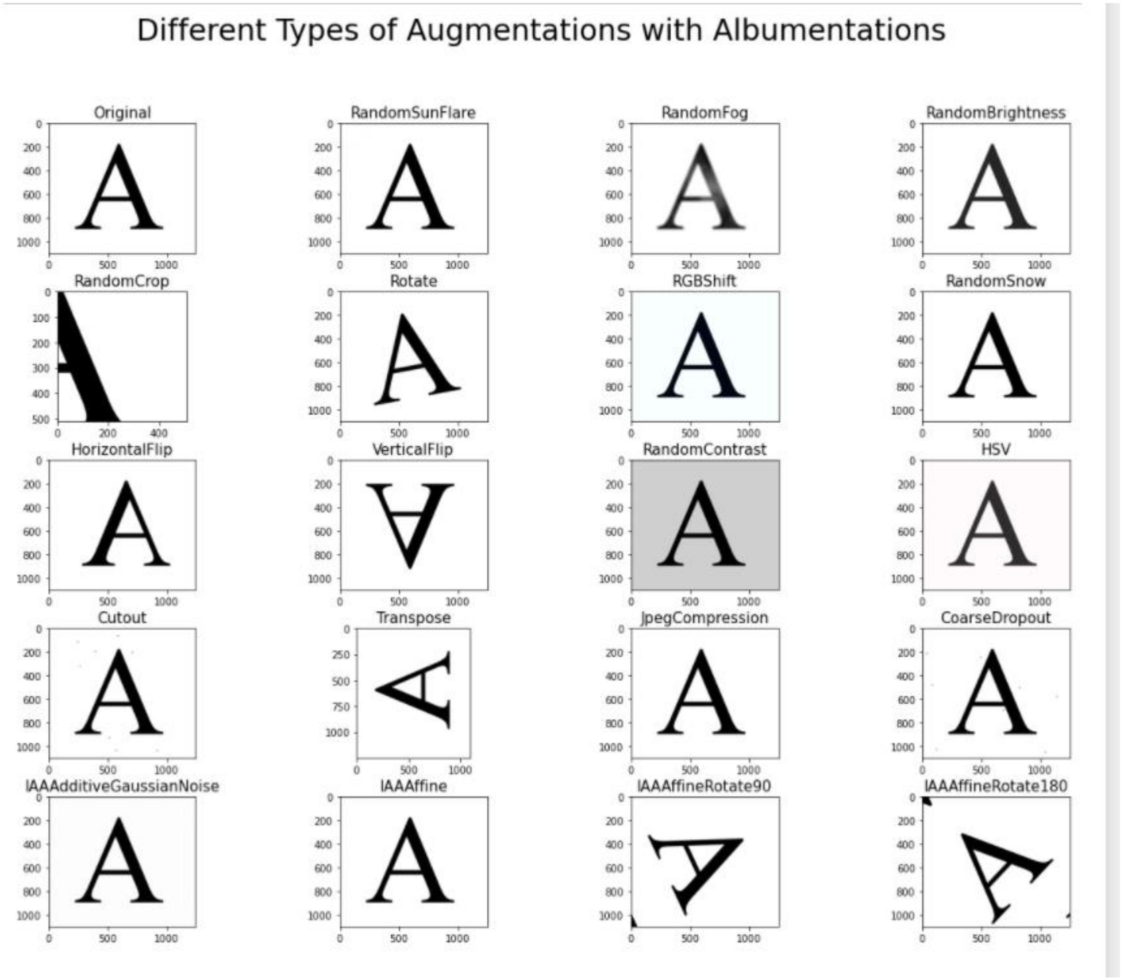


Albumentations Augmentations

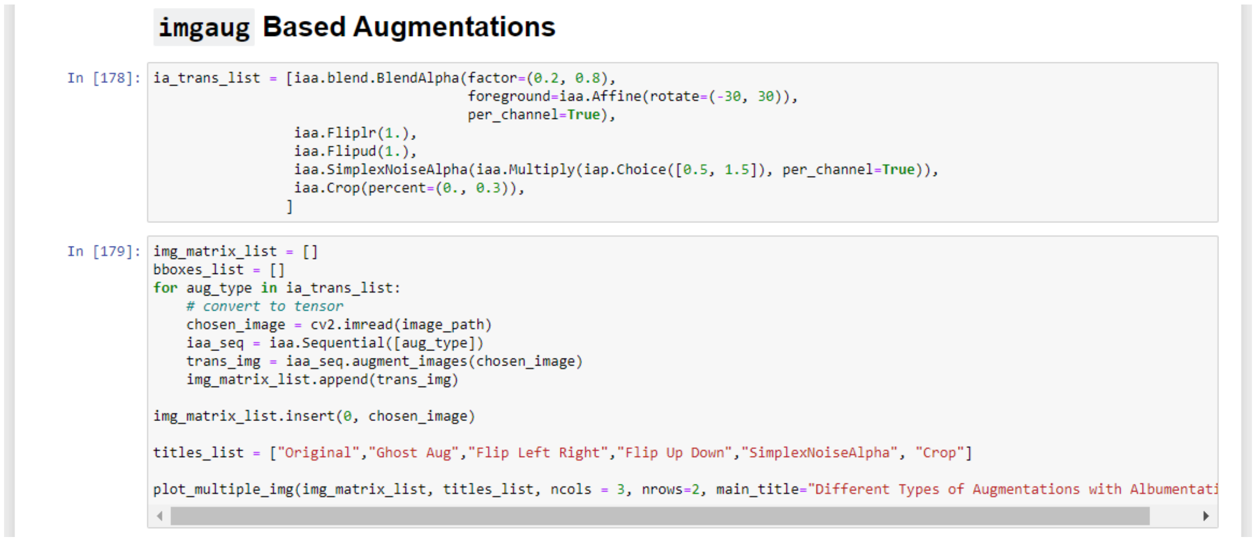
- Albumentations part adapted from my good friend Hongnan's notebbok in the Global Wheat Detection competition (https://www.kaggle.com/reighns/augmentations-data-cleaning-and-bounding-boxes#Bounding-Boxes-with-Albumentations)
- Added more augmentations which may be useful
- Added TensorFlow and Torchvision versions of the augmentations

```
In [171]: albumentation_list = [A.RandomSunFlare(p=1),
                      A.RandomFog(p=1),
                      A.RandomBrightness(p=1),
                      A.RandomCrop(p=1,height = 512, width = 512),
                      A.Rotate(p=1, limit=90),
                      A.RGBShift(p=1),
                      A.RandomSnow(p=1),
                      A.HorizontalFlip(p=1),
                      A.VerticalFlip(p=1),
                      A.RandomContrast(limit = 0.5,p = 1),
                      A.HueSaturationValue(p=1,hue_shift_limit=20, sat_shift_limit=30, val_shift_limit=50),
                      A.Cutout(p=1),
                      A.Transpose(p=1),
                      A.JpegCompression(p=1),
                      A.CoarseDropout(p=1),
                      A.IAAAdditiveGaussianNoise(loc=0, scale=(2.5500000000000003, 12.75), per_channel=False, p=1),
                      A.IAAAffine(scale=1.0, translate_percent=None, translate_px=None, rotate=0.0, shear=0.0, order=1, cval=0, r
                      A.IAAAffine(rotate=90., p=1),
                      A.IAAAffine(rotate=180., p=1)]
```

Picture 9 – Albumentations Augmentations



Picture 10 – result of Albumentations Augmentations imgaug 3 is a library for image augmentation in machine learning experiments. It supports a wide range of augmentation techniques, allows to easily combine these and to execute them in random order or on multiple CPU cores, has a simple yet powerful stochastic interface and can not only augment images, but also keypoints/landmarks, bounding boxes, heatmaps and segmentation maps.

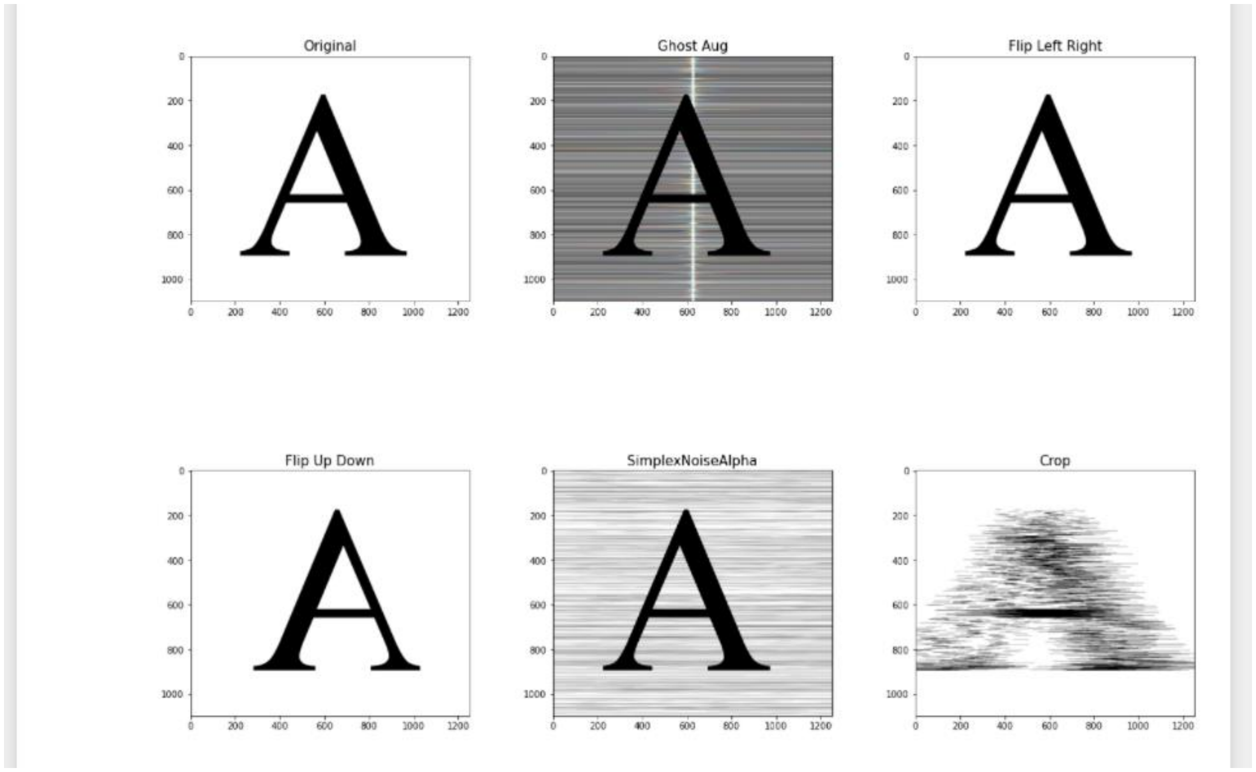


`imgaug` Based Augmentations

```
In [178]: ia_trans_list = [iaa.blend.BlendAlpha(factor=(0.2, 0.8),
                                   foreground=iaa.Affine(rotate=(-30, 30)),
                                   per_channel=True),
                     iaa.Fliplr(1.),
                     iaa.Flipud(1.),
                     iaa.SimplexNoiseAlpha(iaa.Multiply(iap.Choice([0.5, 1.5]), per_channel=True)),
                     iaa.Crop(percent=(0., 0.3)),
                    ]
```

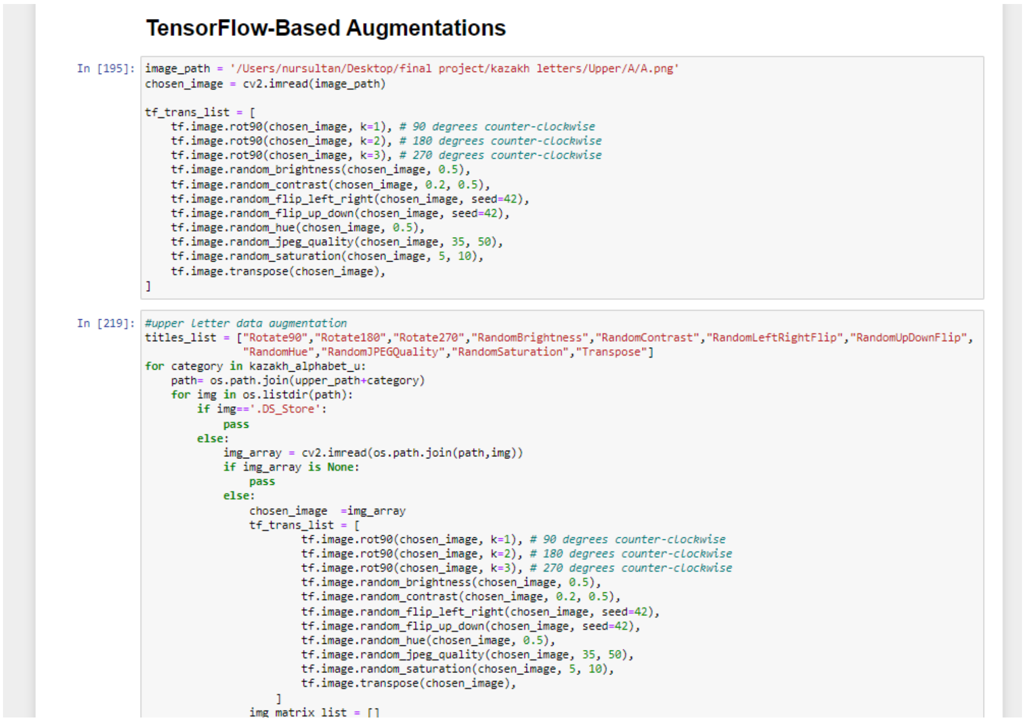
```
In [179]: img_matrix_list = []
bboxes_list = []
for aug_type in ia_trans_list:
    # convert to tensor
    chosen_image = cv2.imread(image_path)
    iaa_seq = iaa.Sequential([aug_type])
    trans_img = iaa_seq.augment_images(chosen_image)
    img_matrix_list.append(trans_img)

img_matrix_list.insert(0, chosen_image)

titles_list = ["Original","Ghost Aug","Flip Left Right","Flip Up Down","SimplexNoiseAlpha", "Crop"]

plot_multiple_img(img_matrix_list, titles_list, ncols = 3, nrows=2, main_title="Different Types of Augmentations with Albumentati
```

Picture 11 – code for imgaug based augmentations



Picture 12 – result of imgaug based augmentations

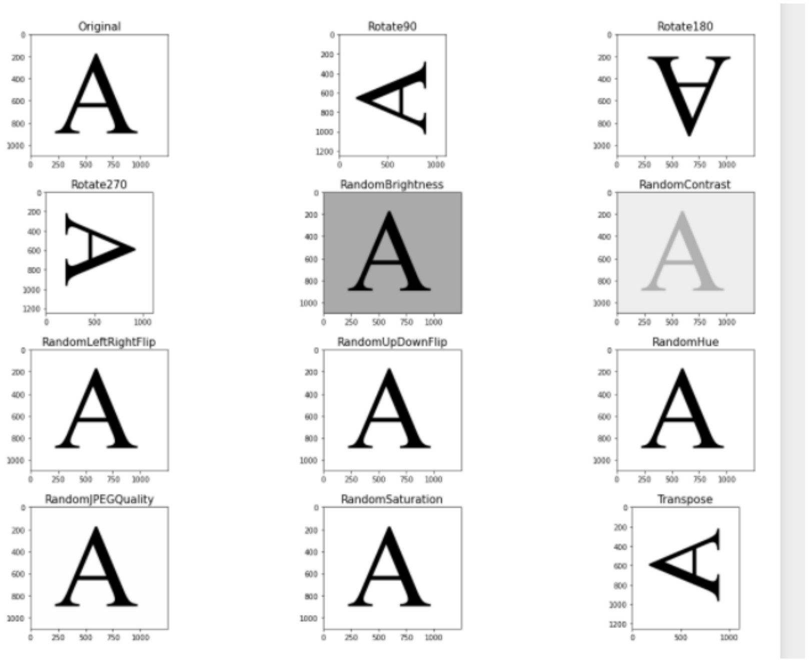
In TensorFlow 4 , data augmentation is accomplished using the ImageDataGenerator class. It is exceedingly simple to understand and to use. The entire dataset is looped over in each epoch, and the images in the dataset are transformed as per the options and values selected.



TensorFlow-Based Augmentations

```
In [195]: image_path = '/Users/nursultan/Desktop/final project/kazakh letters/Upper/A/A.png'
          chosen_image = cv2.imread(image_path)

          tf_trans_list = [
              tf.image.rot90(chosen_image, k=1), # 90 degrees counter-clockwise
              tf.image.rot90(chosen_image, k=2), # 180 degrees counter-clockwise
              tf.image.rot90(chosen_image, k=3), # 270 degrees counter-clockwise
              tf.image.random_brightness(chosen_image, 0.5),
              tf.image.random_contrast(chosen_image, 0.2, 0.5),
              tf.image.random_flip_left_right(chosen_image, seed=42),
              tf.image.random_flip_up_down(chosen_image, seed=42),
              tf.image.random_hue(chosen_image, 0.5),
              tf.image.random_jpeg_quality(chosen_image, 35, 50),
              tf.image.random_saturation(chosen_image, 5, 10),
              tf.image.transpose(chosen_image),
          ]

In [219]: #upper letter data augmentation
          titles_list = ["Rotate90","Rotate180","Rotate270","RandomBrightness","RandomContrast","RandomLeftRightFlip","RandomUpDownFlip",
                         "RandomHue","RandomJPEGQuality","RandomSaturation","Transpose"]
          for category in kazakh_alphabet_u:
              path= os.path.join(upper_path+category)
              for img in os.listdir(path):
                  if img=='.DS_Store':
                      pass
                  else:
                      img_array = cv2.imread(os.path.join(path,img))
                      if img_array is None:
                          pass
                      else:
                          chosen_image  =img_array
                          tf_trans_list = [
                                  tf.image.rot90(chosen_image, k=1), # 90 degrees counter-clockwise
                                  tf.image.rot90(chosen_image, k=2), # 180 degrees counter-clockwise
                                  tf.image.rot90(chosen_image, k=3), # 270 degrees counter-clockwise
                                  tf.image.random_brightness(chosen_image, 0.5),
                                  tf.image.random_contrast(chosen_image, 0.2, 0.5),
                                  tf.image.random_flip_left_right(chosen_image, seed=42),
                                  tf.image.random_flip_up_down(chosen_image, seed=42),
                                  tf.image.random_hue(chosen_image, 0.5),
                                  tf.image.random_jpeg_quality(chosen_image, 35, 50),
                                  tf.image.random_saturation(chosen_image, 5, 10),
                                  tf.image.transpose(chosen_image),
                              ]
                          img_matrix_list = []
```

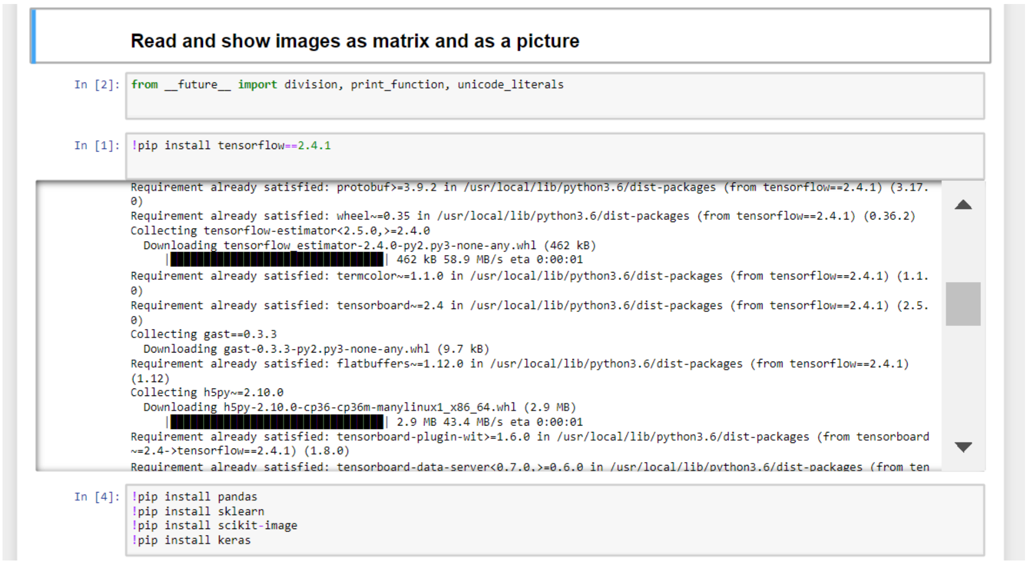
Picture 13 – code for TensorFlow-Based Augmentations



Picture 14 - TensorFlow-Based Augmentations result

3. Model Training

In literature review we found different research approaches in Optical text recognition for many languages but besides the Kazakh language. Including this fact, we made a decision to create our methodology for Kazakh Language in OCR. Our idea is separate every row into words, then these words to characters. After partition steps we recognize every character in trained model on VGG16 architectures. This architecture integrated and trained on every character especially 42 kazakh letters both in uppercase and lowercase. After we trained a model, we estimated performance of our architecture on these metrics such as Accuracy, Mis-Classification, Sensitivity, Specificity, Precision, F1 Score, MCC.

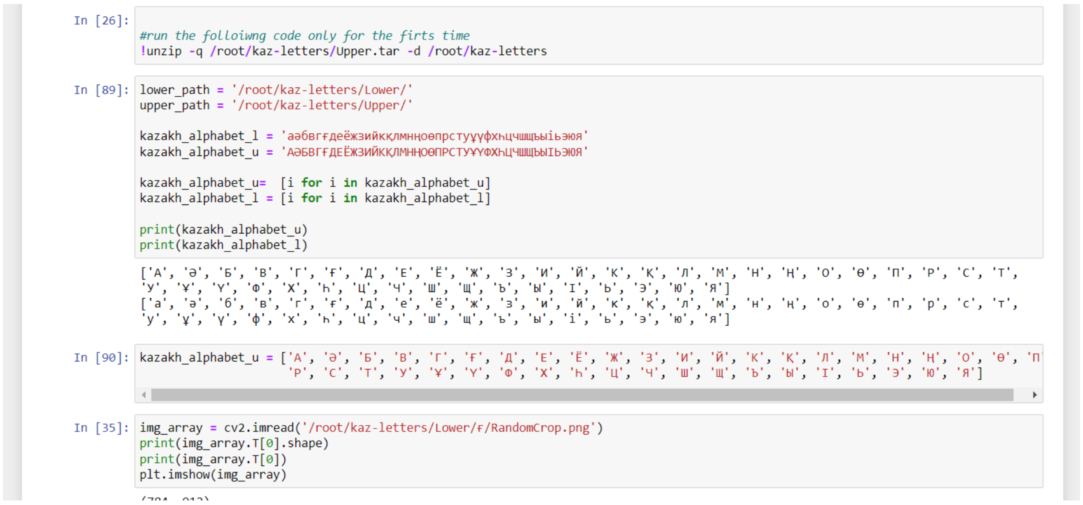


Read and show images as matrix and as a picture

```
In [2]: from __future__ import division, print_function, unicode_literals

In [1]: !pip install tensorflow==2.4.1

Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.4.1) (3.17.
0)
Requirement already satisfied: wheel~=0.35 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.4.1) (0.36.2)
Collecting tensorflow-estimator<2.5.0,>=2.4.0
  Downloading tensorflow_estimator-2.4.0-py2.py3-none-any.whl (462 kB)
     |████████████████████████████████| 462 kB 58.9 MB/s eta 0:00:01
Requirement already satisfied: termcolor~=1.1.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.4.1) (1.1.
0)
Requirement already satisfied: tensorboard~=2.4 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.4.1) (2.5.
0)
Collecting gast==0.3.3
  Downloading gast-0.3.3-py2.py3-none-any.whl (9.7 kB)
Requirement already satisfied: flatbuffers~=1.12.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.4.1)
(1.12)
Collecting h5py~=2.10.0
  Downloading h5py-2.10.0-cp36-cp36m-manylinux1_x86_64.whl (2.9 MB)
     |████████████████████████████████| 2.9 MB 43.4 MB/s eta 0:00:01
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.6/dist-packages (from tensorboard
~=2.4->tensorflow==2.4.1) (1.8.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.6/dist-packages (from ten

In [4]: !pip install pandas
        !pip install sklearn
        !pip install scikit-image
        !pip install keras
```

Picture 15 – Importing necessary libraries

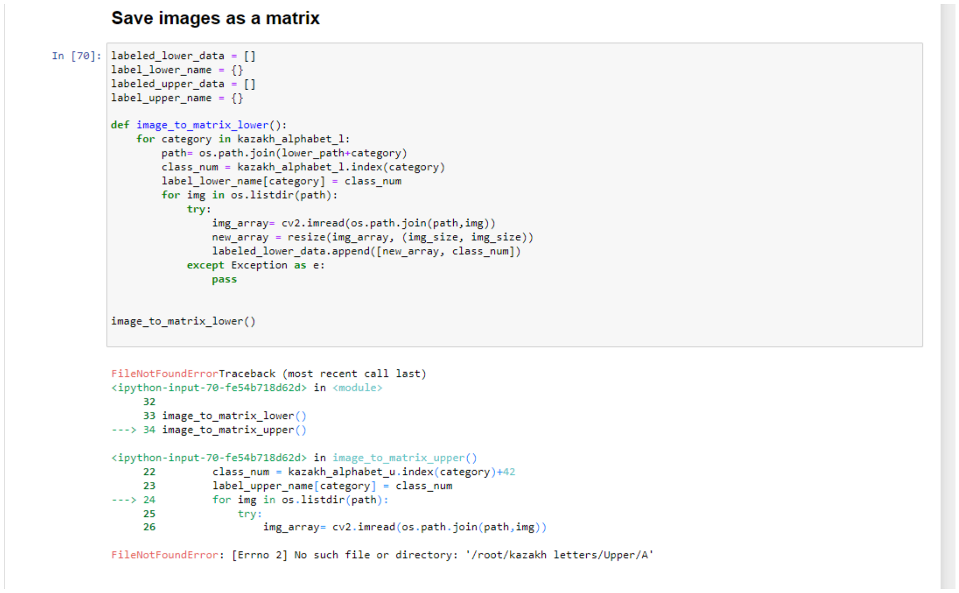
Firstly, we imported all necessary libraries for training our model and all necessary packages.



```
In [26]:
         #run the folloiwng code only for the firts time
         !unzip -q /root/kaz-letters/Upper.tar -d /root/kaz-letters

In [89]: lower_path = '/root/kaz-letters/Lower/'
         upper_path = '/root/kaz-letters/Upper/'

         kazakh_alphabet_l = 'аәбвгғдеёжзийкқлмнңоөпрстуұүфхһцчшщъыіьэюя'
         kazakh_alphabet_u = 'АӘБВГҒДЕЁЖЗИЙКҚЛМНҢОӨПРСТУҰҮФХҺЦЧШЩЪЫІЬЭЮЯ'

         kazakh_alphabet_u=  [i for i in kazakh_alphabet_u]
         kazakh_alphabet_l = [i for i in kazakh_alphabet_l]

         print(kazakh_alphabet_u)
         print(kazakh_alphabet_l)

         ['А', 'Ә', 'Б', 'В', 'Г', 'Ғ', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К', 'Қ', 'Л', 'М', 'Н', 'Ң', 'О', 'Ө', 'П', 'Р', 'С', 'Т',
         'У', 'Ұ', 'Ү', 'Ф', 'Х', 'Һ', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'І', 'Ь', 'Э', 'Ю', 'Я']
         ['а', 'ә', 'б', 'в', 'г', 'ғ', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'қ', 'л', 'м', 'н', 'ң', 'о', 'ө', 'п', 'р', 'с', 'т',
         'у', 'ұ', 'ү', 'ф', 'х', 'һ', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'і', 'ь', 'э', 'ю', 'я']

In [90]: kazakh_alphabet_u = ['А', 'Ә', 'Б', 'В', 'Г', 'Ғ', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К', 'Қ', 'Л', 'М', 'Н', 'Ң', 'О', 'Ө', 'П
                              'Р', 'С', 'Т', 'У', 'Ұ', 'Ү', 'Ф', 'Х', 'Һ', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'І', 'Ь', 'Э', 'Ю', 'Я']

In [35]: img_array = cv2.imread('/root/kaz-letters/Lower/ғ/RandomCrop.png')
         print(img_array.T[0].shape)
         print(img_array.T[0])
         plt.imshow(img_array)
```

Picture 16 – Classification of letters

Then, we have classified Kazakh letters to classes. As a result, we came with 84 letter classes for 42 lowercase letters and 42 uppercase letters.



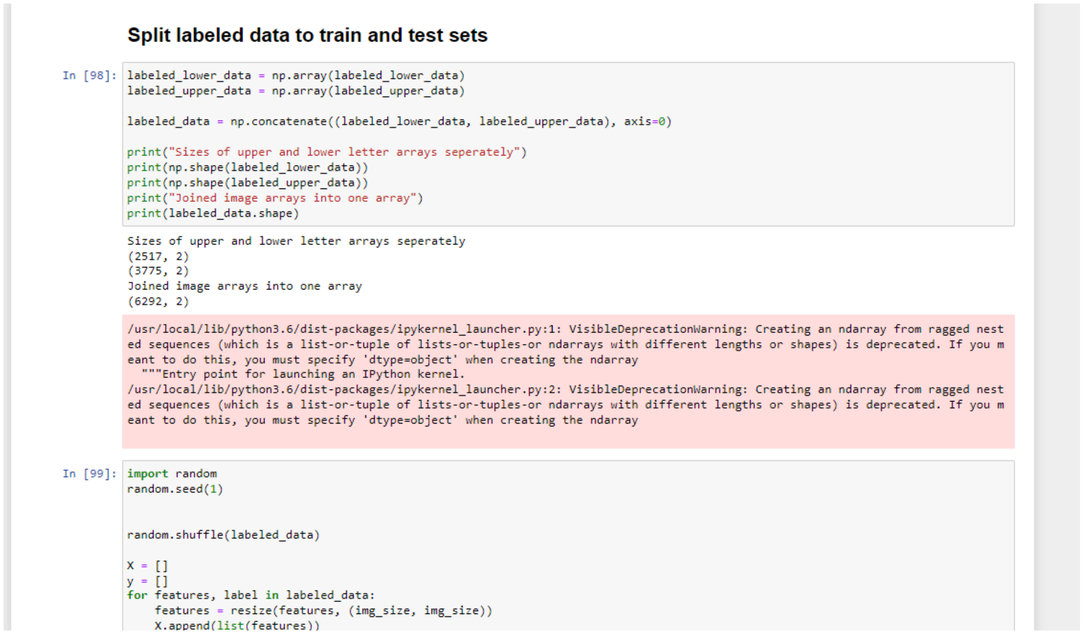
Save images as a matrix

```
In [70]: labeled_lower_data = []
         label_lower_name = {}
         labeled_upper_data = []
         label_upper_name = {}

         def image_to_matrix_lower():
             for category in kazakh_alphabet_l:
                 path= os.path.join(lower_path+category)
                 class_num = kazakh_alphabet_l.index(category)
                 label_lower_name[category] = class_num
                 for img in os.listdir(path):
                     try:
                         img_array= cv2.imread(os.path.join(path,img))
                         new_array = resize(img_array, (img_size, img_size))
                         labeled_lower_data.append([new_array, class_num])
                     except Exception as e:
                         pass


         image_to_matrix_lower()
```

```
FileNotFoundErrorTraceback (most recent call last)
<ipython-input-70-fe54b718d62d> in <module>
     32
     33 image_to_matrix_lower()
---> 34 image_to_matrix_upper()

<ipython-input-70-fe54b718d62d> in image_to_matrix_upper()
     22         class_num = kazakh_alphabet_u.index(category)+42
     23         label_upper_name[category] = class_num
---> 24         for img in os.listdir(path):
     25             try:
     26                 img_array= cv2.imread(os.path.join(path,img))

FileNotFoundError: [Errno 2] No such file or directory: '/root/kazakh letters/Upper/A'
```

Picture 17 – processing image

Then we processed images and saved as a matrix.



Split labeled data to train and test sets

```
In [98]: labeled_lower_data = np.array(labeled_lower_data)
         labeled_upper_data = np.array(labeled_upper_data)

         labeled_data = np.concatenate((labeled_lower_data, labeled_upper_data), axis=0)

         print("Sizes of upper and lower letter arrays seperately")
         print(np.shape(labeled_lower_data))
         print(np.shape(labeled_upper_data))
         print("Joined image arrays into one array")
         print(labeled_data.shape)
```

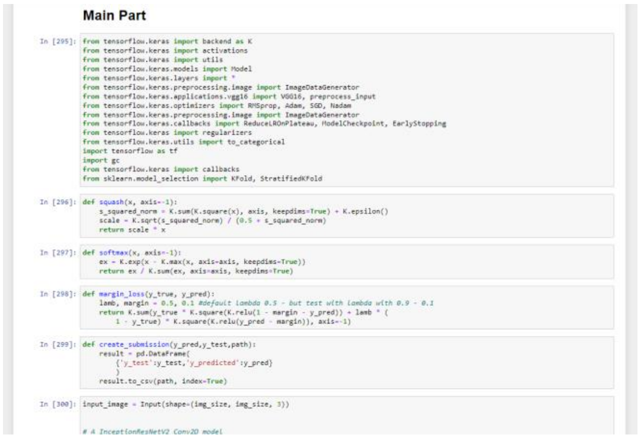
```
Sizes of upper and lower letter arrays seperately
(2517, 2)
(3775, 2)
Joined image arrays into one array
(6292, 2)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: VisibleDeprecationWarning: Creating an ndarray from ragged nest
ed sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you m
eant to do this, you must specify 'dtype=object' when creating the ndarray
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: VisibleDeprecationWarning: Creating an ndarray from ragged nest
ed sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you m
eant to do this, you must specify 'dtype=object' when creating the ndarray
```

```
In [99]: import random
         random.seed(1)


         random.shuffle(labeled_data)

         X = []
         y = []
         for features, label in labeled_data:
             features = resize(features, (img_size, img_size))
             X.append(list(features))
```

Picture 18 – splitting data for train and test cases

After we processed images we have split the data for train case and test case. And created two arrays, first is for saving image matrix and second one for storing result class of this matrix.

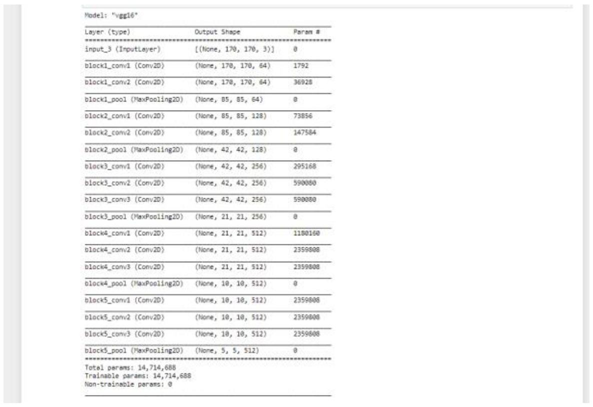


Main Part

```
In [295]: from tensorflow.keras import backend as K
          from tensorflow.keras import activations
          from tensorflow.keras import utils
          from tensorflow.keras.models import Model
          from tensorflow.keras.layers import *
          from tensorflow.keras.preprocessing.image import ImageDataGenerator
          from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
          from tensorflow.keras.optimizers import RMSprop, Adam, SGD, Nadam
          from tensorflow.keras.preprocessing.image import ImageDataGenerator
          from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint, EarlyStopping
          from tensorflow.keras import regularizers
          from tensorflow.keras.utils import to_categorical
          import tensorflow as tf
          import gc
          from tensorflow.keras import callbacks
          from sklearn.model_selection import KFold, StratifiedKFold

In [296]: def squash(x, axis=-1):
              s_squared_norm = K.sum(K.square(x), axis, keepdims=True) + K.epsilon()
              scale = K.sqrt(s_squared_norm) / (0.5 + s_squared_norm)
              return scale * x

In [297]: def softmax(x, axis=-1):
              ex = K.exp(x - K.max(x, axis=axis, keepdims=True))
              return ex / K.sum(ex, axis=axis, keepdims=True)

In [298]: def margin_loss(y_true, y_pred):
              lamb, margin = 0.5, 0.1 #default lambda 0.5 - but test with lambda with 0.9 - 0.1
              return K.sum(y_true * K.square(K.relu(1 - margin - y_pred)) + lamb * (
                  1 - y_true) * K.square(K.relu(y_pred - margin)), axis=-1)

In [299]: def create_submission(y_pred,y_test,path):
              result = pd.DataFrame(
                  {'y_test':y_test,'y_predicted':y_pred}
                  )
              result.to_csv(path, index=True)

In [300]: input_image = Input(shape=(img_size, img_size, 3))


          # A InceptionResNetV2 Conv2D model
```

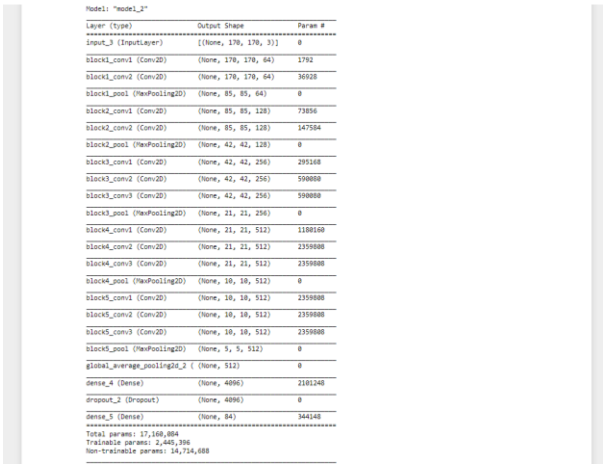
Picture 19 – main part

After we separated our data, we started to train our model on that data using transfer learning technique.



```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 170, 170, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 170, 170, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 170, 170, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 85, 85, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 85, 85, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 85, 85, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 42, 42, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 42, 42, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 42, 42, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 42, 42, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 21, 21, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 21, 21, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 21, 21, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 21, 21, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 10, 10, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 5, 5, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Picture 20 – VGG16 model



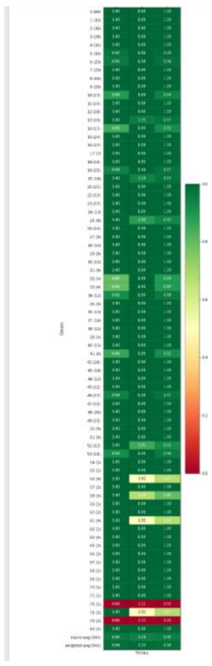
```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 170, 170, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 170, 170, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 170, 170, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 85, 85, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 85, 85, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 85, 85, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 42, 42, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 42, 42, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 42, 42, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 42, 42, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 21, 21, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 21, 21, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 21, 21, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 21, 21, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 10, 10, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 5, 5, 512)         0
_________________________________________________________________
global_average_pooling2d_2 ( (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 4096)              2101248
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_5 (Dense)              (None, 84)                344148
=================================================================
Total params: 17,160,084
Trainable params: 2,445,396
Non-trainable params: 14,714,688
_________________________________________________________________
```

Picture 21 – VGG16 model

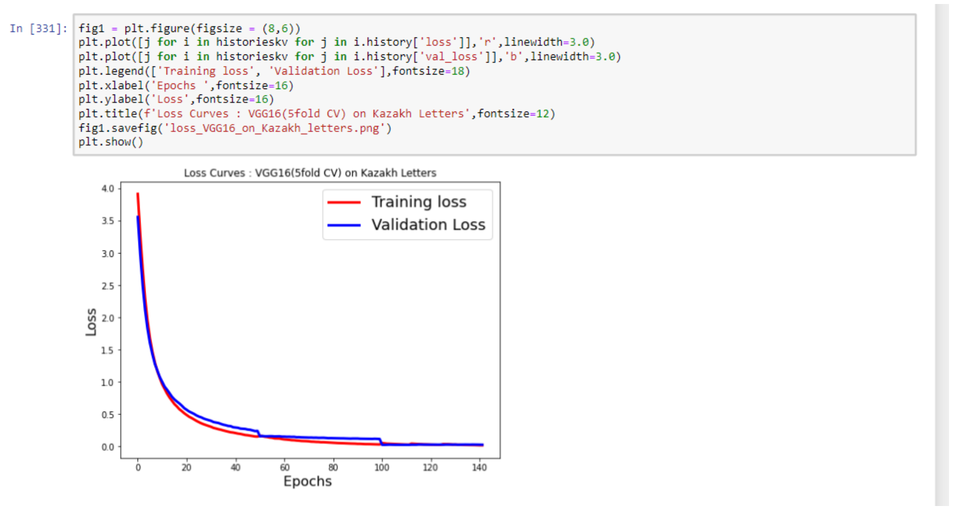
The Visual Geometry Group (VGG) was the runner up of ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2014. The main contribution of this model shows that the depth (i.e., number of layers) of a convolutional neural network is the critical component of high recognition or classification accuracy. In this architecture, two convolutional layers are used consecutively with a rectified linear unit (ReLU) activation function followed by single max-pooling layer, several fully connected layers with ReLU and soft-max as the final layer. The 3×3 convolutional filters with stride 2 is applied for performing filtering and sub-sampling operations simultaneously in VGG-E version. There are three types of VGGNet based on the architecture. These three network contains 11, 16 and 19 layers and named as VGG-11, VGG-16 and VGG-19, respectively. There are five convolution layers, three max-pooling layers and three fully connected (FC) layers. The configuration of VGG16 is as follows: number of convolution layers: 16, fully connected layers: 3, weights: 138 Million and Multiplication and Accumulates (MACs): 15.5G. In this architecture, the multiple convolutional layers are incorporated which is followed by a max-pooling layer. In this implementation, we have used VGG16 with less number of feature maps in each convolutional block compared to standard VFF16 model.

In our model we used VGG16 architecture and the result were very high.

4. Results

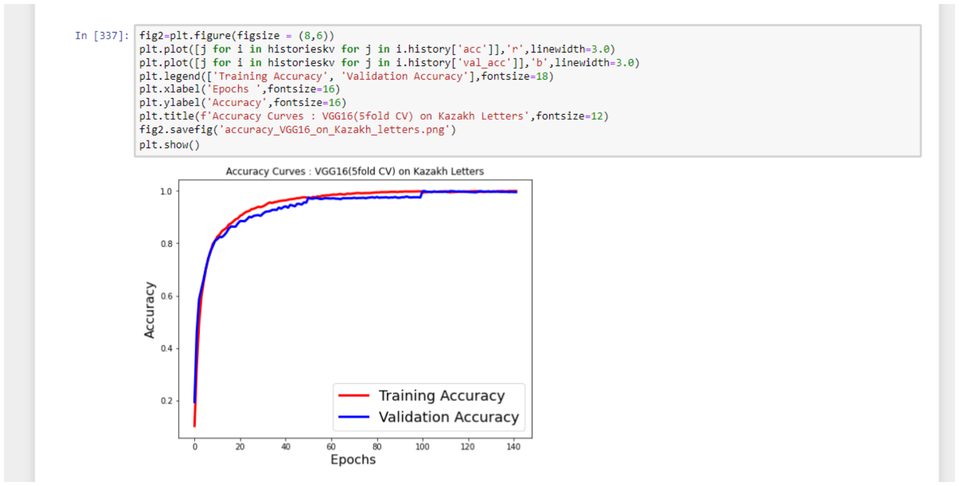


Picture 22 – Results of Precision, Recall, F1 Score of 84 letter classes



```
In [331]: fig1 = plt.figure(figsize = (8,6))
          plt.plot([j for i in historieskv for j in i.history['loss']],'r',linewidth=3.0)
          plt.plot([j for i in historieskv for j in i.history['val_loss']],'b',linewidth=3.0)
          plt.legend(['Training loss', 'Validation Loss'],fontsize=18)
          plt.xlabel('Epochs ',fontsize=16)
          plt.ylabel('Loss',fontsize=16)
          plt.title(f'Loss Curves : VGG16(5fold CV) on Kazakh Letters',fontsize=12)
          fig1.savefig('loss_VGG16_on_Kazakh_letters.png')
          plt.show()
```

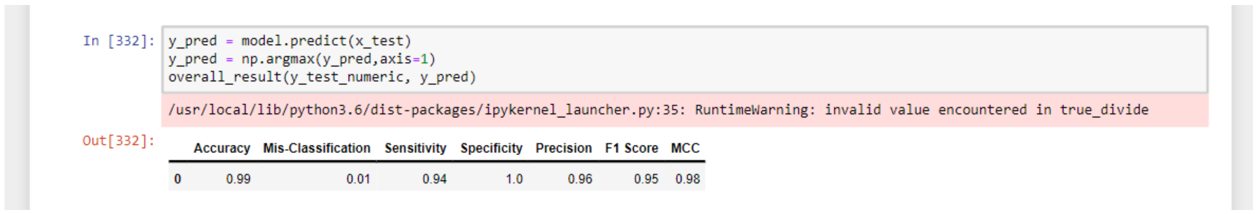
Picture 23 – Loss curve on learning



```
In [337]: fig2=plt.figure(figsize = (8,6))
          plt.plot([j for i in historieskv for j in i.history['acc']],'r',linewidth=3.0)
          plt.plot([j for i in historieskv for j in i.history['val_acc']],'b',linewidth=3.0)
          plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=18)
          plt.xlabel('Epochs ',fontsize=16)
          plt.ylabel('Accuracy',fontsize=16)
          plt.title(f'Accuracy Curves : VGG16(5fold CV) on Kazakh Letters',fontsize=12)
          fig2.savefig('accuracy_VGG16_on_Kazakh_letters.png')
          plt.show()
```

Picture 24 – Accuracy curve on learning

This is learning curve of our model. As we can clearly seen from the diagrams that learning curves are very good trained and Training and Validation curves aren't differ so much



```
In [332]: y_pred = model.predict(x_test)
          y_pred = np.argmax(y_pred,axis=1)
          overall_result(y_test_numeric, y_pred)

          /usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:35: RuntimeWarning: invalid value encountered in true_divide
```

Out[332]:

	Accuracy	Mis-Classification	Sensitivity	Specificity	Precision	F1 Score	MCC
0	0.99	0.01	0.94	1.0	0.96	0.95	0.98

Picture 25 – Final results

In order to evaluate the accuracy of the models performance we used metrics such as accuracy, sensitivity/recall, specificity, precision, F1 score, MCC. A VGG16 architecture gave us high performance results.. The overall results can be seen in the table above.

Conclusion

In this project Kazakh recognition system has been developed with its own Kazakh letters dataset created by data augmentation method. We generated dataset of Kazakh letters for training and trained and tested network for OCR. We hope that our work will help many people in retrieving information from text image in Kazakh language. Like students who study in Kazakh and in many cases they have to retype some text from image format to editable text. So, in the future we can make user-friendly interface application based on this work.