Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по предмету:
«Машинное обучение»
По лабораторной работе №6
«Применение сверточных нейронных сетей (многоклассовая классификация)»

Выполнил: Сенькович Дмитрий Сергеевич
магистрант кафедры информатики
группы №858642

Проверил: Стержанов Максим Валерьевич
доцент, кандидат технических наук

Минск 2020

Оглавление

1 Постановка задачи

Имеются тренировочная и тестовая выборки изображений жестов букв. Требуется построить многоклассовый классификатор на основе нейронной сети для распознавания жестов. Изображения представлены в формате csv, содержащим метку - номер буквы - и пиксели изображения в оттенках серого.

2 Чтение данных и формирование выборок

Покажем, как будем преобразовывать, считывать данные и дополнять данные:

```
def augment_vertical_flip():
    if os.path.isfile(AUGMENTED_VERTICAL_FLIP_FILE_NAME):
        return


    original_dataset, labels = load_csv(TRAIN_DATASET_FILE_NAME)
    augmented_dataset = []
    for image in original_dataset:
        augmented_dataset.append(np.flipud(image).ravel())
    augmented_dataset = np.array(augmented_dataset)
    #show_image(augmented_dataset[0].reshape(28, 28))
    augmented = np.column_stack([labels, augmented_dataset])
    np.savetxt(AUGMENTED_VERTICAL_FLIP_FILE_NAME, augmented, delimiter=',',
fmt='%d')


def augment_horizontal_flip():
    if os.path.isfile(AUGMENTED_HORIZONTAL_FLIP_FILE_NAME):
        return


    original_dataset, labels = load_csv(TRAIN_DATASET_FILE_NAME)
    augmented_dataset = []
    for image in original_dataset:
        augmented_dataset.append(np.fliplr(image).ravel())
    augmented_dataset = np.array(augmented_dataset)
```

```
    #show_image(augmented_dataset[0].reshape(28, 28))

    augmented = np.column_stack([labels, augmented_dataset])

    np.savetxt(AUGMENTED_HORIZONTAL_FLIP_FILE_NAME, augmented, delimiter=',',
fmt='%d')




def augment_gaussian_noise():

    if os.path.isfile(AUGMENTED_GUASSIAN_NOISE_FLIP_FILE_NAME):

        return



    original_dataset, labels = load_csv(TRAIN_DATASET_FILE_NAME)

    augmented_dataset = []

    for image in original_dataset:

        augmented_dataset.append(with_gaussian_noise(image).ravel())
```

Чтение данных и формирование выборок практически такое же, как и в прошлой лабораторной работе. Самое важное изменения - формат данных. Данные в формате csv считываются и записываются с помощью библиотеки pandas. Также, теперь мы не будем записывать предсказанные сетью классы в отдельный файл, а будем проверять точность классификатора на тестовой выборке сразу.

3 Архитектура и результаты для собственной сети

Будем использовать почти такую же сеть, как и в прошлой лабораторной работе, за исключением выходного слоя: ввиду того, что в лабораторной работе идет построение много классового классификатора, выходной уровень будет теперь содержать столько нейронов, сколько имеется классов:

```
keras.layers.Dense(CLASSES_COUNT, activation='softmax')
```

Для собственной архитектуры сети результаты получились следующие:

Данные	Accuracy (validation, %)	Accuracy (test, %)
Без augmentation	100	98.84
noise	100	99.28
horizontal flip	99.99	99.56
vertical flip	100	98.53

4 Готовые сети

В лабораторной работе предлагается использовать и сравнить результаты с какой-нибудь готовой сетью. Так же, как и в предыдущей работе, была выбрана сеть VGG16. Способ использования сети абсолютно аналогичен использованию в предыдущей лабораторной работе. Но, стоит заметить, что при адаптации решения к лабораторной работе возникли следующие трудности:

- VGG16 и остальные сети работают на изображениях с изображениями не меньше 32x32. Ввиду того, что наши изображения имеют разрешение 28x28, пришлось изменить их размер:

```
def upscale(dataset_filename, output_dataset_filename):

    if os.path.isfile(output_dataset_filename):

        return



    original_dataset, labels = load_csv(dataset_filename, 28)

    upscaled_dataset = []

    for image in original_dataset:

        upscaled_dataset.append(upscale_image(image).ravel())

    upscaled_dataset = np.array(upscaled_dataset)

    #show_image(augmented_dataset[0].reshape(28, 28))

    upscaled = np.column_stack([labels, upscaled_dataset])

    np.savetxt(output_dataset_filename, upscaled, delimiter=',',
fmt='%d')




def upscale_image(image):
```

```
    return cv2.resize(image.astype('float32'), (32, 32),
interpolation=cv2.INTER_AREA)
```

Далее логика построения выборок практически такая же, как и с сетью с собственной архитектурой.

- Готовые сети работают с цветными изображениями. Для симуляции цветного изображения было решено просто продублировать изображение в трехканальное с одинаковыми компонентами в каждом канале:

```
def to_fake_rgb(dataset):

    return np.repeat(dataset, 3, -1)
```

Как и в прошлой лабораторной работе, заменим существующий выходной слой на softmax слой.

Покажем результаты обучения данной сети:

Данные	Accuracy (validation, %)	Accuracy (test, %)
Без augmentation	100	99.97
noise	100	99.70
horizontal flip	100	100
vertical flip	100	99.97

Главным вопросом лабораторной работы было понять, нужно ли была здесь вообще передаточное обучение. На мой взгляд, можно было бы обойтись и без него, поскольку результаты очень хорошие и на сети с собственной архитектурой. С другой стороны, появляются подозрения, что тестовая выборка не так хороша для оценивания точности классификатора и/или данные очень просты, так как точность чересчур высока. Согласно экспериментам, лучший результат получился в случае отображения изображений по горизонтали с использованием сети VGG16.