ПРИКЛАДНА МАТЕМАТИКА

УДК 004.891 https://doi.org/10.31713/vt2202413

Климюк Ю. Є., к.т.н., доцент (Національний університет водного господарства та природокористування, м. Рівне, yu.ye.klymiuk@nuwm.edu.ua)

РОЗРОБКА КЛАСИФІКАТОРА ЗОБРАЖЕНЬ ОЦІНОК ДЛЯ РОЗПІЗНАВАННЯ ОЦІНОК ІЗ СКАНКОПІЙ ДОДАТКІВ ДО СВІДОЦТВ ПРО ЗДОБУТТЯ ПОВНОЇ ЗАГАЛЬНОЇ СЕРЕДНЬОЇ ОСВІТИ АБІТУРІЄНТІВ

Розробку класифікатора зображень оцінок для розпізнавання оцінок з додатків до свідоцтв про здобуття повної загальної середньої освіти абітурієнтів реалізовано мовою програмування Python з використанням редактора VS Code та розширення Jupyter Notebook шляхом донавчання моделі ResNet50, попередньо навченої на наборі даних ImageNet, за допомогою бібліотеки PyTorch на основі згенерованого набору даних, що містить 1881 тренувальних, 331 перевірочних та 338 тестових зображень оцінок, а для обробки зображень використано бібліотеку OpenCV. Донавчання моделі проведено для 50 епох, найкращою виявилась 33. Точність отриманої моделі стосовно розпізнавання зображень оцінок на тестовому наборі даних склала 93,79%.

Розроблений класифікатор при його донавчанні за рахунок урізноманітнення тренувальних, перевірочних та тестових наборів зображень оцінок дозволить розробити додаток, який полегшить абітурієнтам здійснювати розрахунок середнього балу свідоцтва про здобуття повної загальної середньої освіти, а працівникам приймальної комісії перевірку правильності його розрахунку на основі наданої сканкопії зображення додатку до свідоцтва.

Ключові слова: **класифікатор; оцінка; зображення; модель; розпізнавання; свідоцтво; атестат; абітурієнт.**

Вступ. Кожного року однією з найбільш трудомістких операцій для абітурієнтів при подачі заяв для вступу на навчання до закладів вищої освіти в електронній формі онлайн через електронні кабінети є визначення середнього балу свідоцтва (атестата) про здобуття повної загальної середньої освіти, а для працівників приймальних комісій при обробці поданих абітурієнтами заяв – перевірка

коректності його підрахунку [1; 2]. Тому досить актуальною задачею залишається розробка додатку для розрахунку середнього балу свідоцтва про здобуття повної загальної середньої освіти абітурієнтів на основі сканованої копії (фотокопії) зображення додатку. Для створення такого додатку необхідно сформувати тренувальні, перевірочні та тестові набори даних зображень оцінок та розробити класифікатор зображень оцінок. Для реалізації подібних класифікаторів зображень останнім часом популярним підходом є донавчання моделі ResNet50, попередньо навченої на наборі даних ImageNet, за допомогою бібліотеки PyTorch [3–6], а для обробки зображень однією із популярних бібліотек є OpenCV [7–10].

Постановка задачі. Для розробки класифікатора зображень оцінок шляхом донавчання попередньо навченої моделі ResNet50 на основі тренувальних, перевірочних та тестових наборів даних зображень оцінок, який буде використовуватися для розпізнавання оцінок зі сканованих копій (фотокопій) додатків до свідоцтв про здобуття повної загальної середньої освіти абітурієнтами, необхідно:

– розробити код для підготовки додатків до свідоцтв про здобуття повної загальної середньої освіти абітурієнтами із різними допустимими оцінками;

– сформувати додатки до свідоцтв про здобуття повної загальної середньої освіти абітурієнтами із різними допустимими оцінками;

– розробити код для формування тренувальних, перевірочних та тестових наборів даних зображень оцінок на основі сформованих додатків до свідоцтв про здобуття повної загальної середньої освіти абітурієнтами із різними допустимими оцінками;

– сформувати тренувальні, перевірочні та тестові набори даних зображень оцінок;

– розробити класифікатор зображень оцінок шляхом донавчання моделі ResNet50 на основі сформованих тренувальних, перевірочних та тестових наборів даних зображень оцінок.

Алгоритм розв'язання. Для реалізації класифікатора зображень оцінок використано редактор VS Code та розширення Jupyter Notebook і за їх допомогою створено додаток calculation_set_values.ipynb мовою програмування Python.

У додатку реалізовано функцію generation_dataset_grades() (лістинг 1), за допомогою якої на основі стандартного бланка додатка до свідоцтва про здобуття повної загальної середньої освіти розміру A5 attachment.jpg генерується набір зображень додатків до свідоцтв attachment_0_0.png, attachment_0_1.png, ..., із різними допустимими

оцінками.

Лістинг 1. Функція для формування набору зображень додатків до свідоцтв про здобуття повної загальної середньої освіти абітурієнтів із різними допустимими оцінками

```
def generation_dataset_grades(folder_attachments="attachment",
grades=["", "один", "два", "три", "чотири", "п’ять", "шість", "сім", "вісім",
"дев’ять", "десять", "одинадцять", "дванадцять"]):
  '''
  The function of generating a set of images of possible grades of
attachments to certificates
  Parameters
    folder_attachments: Folder with generated attachments to
certificates (default="attachment")
    grades: The list of possible grades in the attachments to the
certificates
    (default=["", "один", "два", "три", "чотири", "п’ять", "шість", "сім",
"вісім",
    "дев’ять", "десять", "одинадцять", "дванадцять"])
  Returns
    folder_dataset: Folder with generated certificate attachments
    A folder with a generated set of images of possible grades of
attachments to certificates
  '''
  folder_dataset_grades = "dataset_grades"
  if os.path.exists(folder_dataset_grades):
    shutil.rmtree(folder_dataset_grades)
  os.makedirs(folder_dataset_grades)
  folders = []
  folders.append(os.path.join(folder_dataset_grades, ‘train’))
  folders.append(os.path.join(folder_dataset_grades, ‘valid’))
  folders.append(os.path.join(folder_dataset_grades, ‘test’))
  for folder in folders:
    os.makedirs(folder)
    for index in range(len(grades)):
      os.makedirs(os.path.join(folder, str(index)))
  for shift in range(5):
    for index in range(len(grades)):
      name = folder_attachments + "_" + str(shift) + "_" + str(index)
      file_name = name + ".png"
      image = cv2.imread(os.path.join(folder_attachments, file_name))
      height = 43
      length = 240
      x = [540, 1335, 1335]
```

```
        y = [552, 85, 807]
        line_count = [12, 12, 6]
        shift_line = 0
        for k in range(3):
          for line in range(line_count[k]):
            # Crop into patches of size MxN.
            tiles = image[y[k] + line * height : y[k] + (line + 1) * height, x[k] :
x[k] + length]
            # Scaling Down the image 0.935 times specifying a single
scale factor.
            scale_down = 0.935
            tiles = cv2.resize(tiles, None, fx=scale_down, fy=scale_down,
              interpolation=cv2.INTER_LINEAR)
            # Save each patch into file directory.
            if shift // 2 == 1:
              cv2.imwrite(os.path.join(random.choice(folders), str(index),
                "grade_" + name + "_" + str(shift_line + line) + ".png"), tiles)
            else:
              cv2.imwrite(os.path.join(folders[0], str(index),
          "grade_" + name + "_" + str(shift_line + line) + ".png"), tiles)
          shift_line += line_count[k]
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    return folder_dataset_grades
```

Для формування тренувальних, перевірочних та тестових наборів даних зображень оцінок розроблено функцію generation_dataset_grade() (лістинг 2), за допомогою якої на основі згенерованих зображень додатків до свідоцтв про здобуття повної загальної середньої освіти абітурієнтів із різними допустимими оцінками генеруються окремі зображення оцінок та випадковим чином розподіляються по папках test, train, valid.

Лістинг 2. Функція формування тренувальних, перевірочних та тестових наборів даних зображень оцінок

```
def generation_dataset_grades(folder_attachments="attachment", grades=["",
"один", "два", "три", "чотири", "п'ять", "шість", "сім", "вісім", "дев'ять", "десять",
"одинадцять", "дванадцять"]):
  '''
  The function of forming a set of attachments to the certificates of applicants
for obtaining
  a complete general secondary education of images with different permissible
grades
  Parameters
    folder_attachments: Folder with generated attachments to certificates
(default="attachment")
```

```
        grades: The list of possible grades in the attachments to the certificates
        (default=["", "один", "два", "три", "чотири", "п'ять", "шість", "сім", "вісім",
        "дев'ять", "десять", "одинадцять", "дванадцять"])
    Returns
        folder_dataset: Folder with generated certificate attachments
        A folder with a generated set of certificate attachments with different valid
ratings
    '''
    folder_dataset_grades = "dataset_grades"
    if os.path.exists(folder_dataset_grades):
        shutil.rmtree(folder_dataset_grades)
    os.makedirs(folder_dataset_grades)
    folders = []
    folders.append(os.path.join(folder_dataset_grades, 'train'))
    folders.append(os.path.join(folder_dataset_grades, 'valid'))
    folders.append(os.path.join(folder_dataset_grades, 'test'))
    for folder in folders:
        os.makedirs(folder)
        for index in range(len(grades)):
            os.makedirs(os.path.join(folder, str(index)))
    for shift in range(5):
        for index in range(len(grades)):
            name = folder_attachments + "_" + str(shift) + "_" + str(index)
            file_name = name + ".png"
            image = cv2.imread(os.path.join(folder_attachments, file_name))
            height = 43
            length = 240
            x = [540, 1335, 1335]
            y = [552, 85, 807]
            line_count = [12, 12, 6]
            shift_line = 0
            for k in range(3):
                for line in range(line_count[k]):
                    # Crop into patches of size MxN.
                    tiles = image[y[k] + line * height : y[k] + (line + 1) * height, x[k] : x[k] +
length]
                    # Scaling Down the image 0.935 times specifying a single scale
factor.
                    scale_down = 0.935
                    tiles = cv2.resize(tiles, None, fx=scale_down, fy=scale_down,
                        interpolation=cv2.INTER_LINEAR)
                    # Save each patch into file directory.
                    if shift // 2 == 1:
                        cv2.imwrite(os.path.join(random.choice(folders), str(index),
                                "grade_" + name + "_" + str(shift_line + line) + ".png"), tiles)
                    else:
                        cv2.imwrite(os.path.join(folders[0], str(index),
                                 "grade_" + name + "_" + str(shift_line + line) + ".png"), tiles)
                shift_line += line_count[k]
```

```
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    return folder_dataset_grades
```

Далі у додатку реалізовано код (лістинг 3), у якому відбувається формування списку можливих оцінок у додатках до свідоцтв list_grades, створення папки attachment зі згенерованими зображеннями додатків до свідоцтв із різними допустимими оцінками та створення папки dataset_grades зі згенерованими наборами зображень можливих оцінок додатків до свідоцтв, розподілених по вкладених папках test, train, valid.

Лістинг 3. Код для формування тренувальних, перевірочних та тестових наборів даних зображень оцінок

```
# The list of possible grades in the attachments to the certificates.
list_grades = ["", "один", "два", "три", "чотири", "п'ять", "шість", "сім", "вісім", "дев'ять", "десять", "одинадцять", "дванадцять", "зараховано", "звільнено", "звільнена", "звільнений"]
# Generation of attachments to certificates.
folder_attachments = generation_attachments(attachment="attachment.jpg", grades=list_grades)
print("A folder \"{:s}\" was created containing the following files:".format(folder_attachments))
files = os.listdir(folder_attachments)
print(files)
print("A total of {:d} attachments to certificates were generated.".format(len(files)))
# Generating a set of images of possible evaluations of applications to certificates.
folder_generation_dataset_grades = generation_dataset_grades(folder_attachments, grades=list_grades)
print("A folder \"{:s}\" with the following subfolders has been created:"
      .format(folder_generation_dataset_grades))
folders = os.listdir(folder_generation_dataset_grades)
print(folders)
for folder in folders:
    print("The folder \"{:s}\" contains the following subfolders:".format(folder))
    folder_grades = os.listdir(os.path.join(folder_generation_dataset_grades, folder))
    print(folder_grades)
    length_folder = 0
    for folder_grade in folder_grades:
        length_folder += len(os.listdir(os.path.join(folder_generation_dataset_grades, folder,
                                                     folder_grade)))
    print("A total of {:d} grades images were generated.".format(length_folder))
```

Далі у додатку реалізовано код (лістинг 4), за допомогою якого зображення зі сформованих наборів даних з використанням перетворень ToTensor та Normalize перетворюються на тензор і нормалізуються на середнє значення та стандартне відхилення всіх зображень у ImageNet, перш ніж пакети даних використовуватимуться разом для донавчання моделі.

Лістинг 4. Код для нормалізації наборів даних

```
# Applying Transforms to the Data
image_transforms = {
   'train': transforms.Compose([
      transforms.ToTensor(),
      transforms.Normalize([0.485, 0.456, 0.406],
                           [0.229, 0.224, 0.225])
   ]),
   'valid': transforms.Compose([
      transforms.ToTensor(),
      transforms.Normalize([0.485, 0.456, 0.406],
                           [0.229, 0.224, 0.225])
   ]),
   'test': transforms.Compose([
      transforms.ToTensor(),
      transforms.Normalize([0.485, 0.456, 0.406],
                           [0.229, 0.224, 0.225])
   ])
}
```

Далі у додатку реалізовано код (лістинг 5), за допомогою якого здійснюється завантаження сформованих тренувальних, перевірочних та тестових наборів даних зображень оцінок, щоб на їх основі виконати донавчання моделі ResNet50. Спочатку налаштовуються папки train, valid, test для тренування і перевірки та розмір партії, а потім виконується завантаження наборів даних (зображень) за допомогою DataLoader. При цьому визначені вище перетворення застосовуються до зображень у ході їх завантаження, а також здійснюється їх перемішування.

Лістинг 5. Код для завантаження тренувальних, перевірочних та тестових наборів даних зображень оцінок

```
# Load the Data
# Set train and valid folder paths
folder_dataset = folder_generation_dataset_grades
folder_train = os.path.join(folder_dataset, 'train')
folder_valid = os.path.join(folder_dataset, 'valid')
folder_test = os.path.join(folder_dataset, 'test')
# Batch size
bs = 32
# Number of classes
```

```
num_classes = len(os.listdir(folder_valid))  #10#2#257
print(num_classes)
# Load Data from folders
data = {
   'train': datasets.ImageFolder(root=folder_train,
transform=image_transforms['train']),
   'valid': datasets.ImageFolder(root=folder_valid,
transform=image_transforms['valid']),
   'test': datasets.ImageFolder(root=folder_test,
transform=image_transforms['test'])
}
# Get a mapping of the indices to the class names, in order to see the output
classes
# of the test images.
idx_to_class = {v: k for k, v in data['train'].class_to_idx.items()}
print(idx_to_class)
# Size of Data, to be used for calculating Average Loss and Accuracy
train_data_size = len(data['train'])
valid_data_size = len(data['valid'])
test_data_size = len(data['test'])
# Create iterators for the Data loaded using DataLoader module
train_data_loader = DataLoader(data['train'], batch_size=bs, shuffle=True)
valid_data_loader = DataLoader(data['valid'], batch_size=bs, shuffle=True)
test_data_loader = DataLoader(data['test'], batch_size=bs, shuffle=True)
```

Далі у додатку реалізовано код (лістинг 6), за допомогою якого здійснюється завантаження та налаштування моделі ResNet50.

Лістинг 6. Код для завантаження та налаштування моделі ResNet50

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(train_data_size, valid_data_size, test_data_size)

# Load pretrained ResNet50 Model
resnet50 = models.resnet50(pretrained=True)
resnet50 = resnet50.to(device)

# Freeze model parameters
for param in resnet50.parameters():
   param.requires_grad = False

# Change the final layer of ResNet50 Model for Transfer Learning
fc_inputs = resnet50.fc.in_features
resnet50.fc = nn.Sequential(
   nn.Linear(fc_inputs, 256),
   nn.ReLU(),
   nn.Dropout(0.4),
   nn.Linear(256, num_classes), # Since 10 possible outputs
   nn.LogSoftmax(dim=1) # For using NLLLoss()
)
# Convert model to be used on GPU
resnet50 = resnet50.to(device)
```

```
# Define Optimizer and Loss Function
loss_func = nn.NLLLoss()
optimizer = optim.Adam(resnet50.parameters())
```

Далі у додатку реалізовано функцію train_and_validate() (лістинг 7), за допомогою якої здійснюється донавчання моделі ResNet50.

Лістинг 7. Функція для донавчання моделі ResNet50

```
def train_and_validate(model, loss_criterion, optimizer, epochs=30):
    '''
    Function to train and validate
    Parameters
        :param model: Model to train and validate
        :param loss_criterion: Loss Criterion to minimize
        :param optimizer: Optimizer for computing gradients
        :param epochs: Number of epochs (default=30)
    Returns
        model: Trained Model with best validation accuracy
        history: (dict object): Having training loss, accuracy and validation loss,
accuracy
        best_epoch: Number of the best epoch
    '''
    start = time.time()
    history = []
    best_loss = 100000.0
    best_epoch = None
    for epoch in range(epochs):
        epoch_start = time.time()
        # Set to training mode
        model.train()
        # Loss and Accuracy within the epoch
        train_loss = 0.0
        train_acc = 0.0
        valid_loss = 0.0
        valid_acc = 0.0
        for i, (inputs, labels) in enumerate(train_data_loader):
            inputs = inputs.to(device)
            labels = labels.to(device)
            # Clean existing gradients
            optimizer.zero_grad()
            # Forward pass - compute outputs on input data using the model
            outputs = model(inputs)
            # Compute loss
            loss = loss_criterion(outputs, labels)
            # Backpropagate the gradients
            loss.backward()
            # Update the parameters
            optimizer.step()
            # Compute the total loss for the batch and add it to train_loss
```

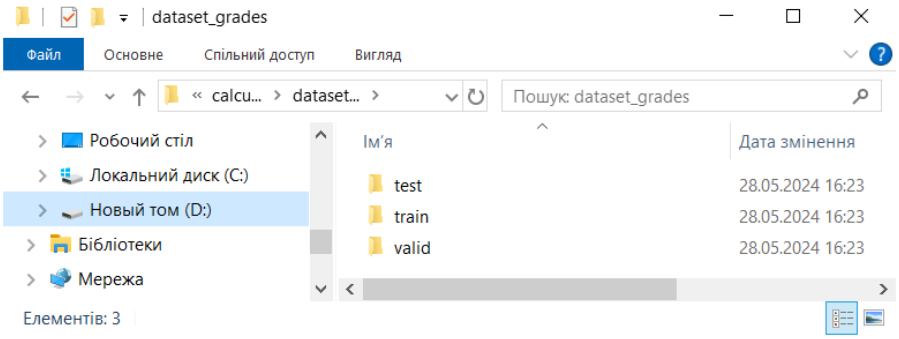
```
            train_loss += loss.item() * inputs.size(0)
            # Compute the accuracy
            ret, predictions = torch.max(outputs.data, 1)
            correct_counts = predictions.eq(labels.data.view_as(predictions))
            # Convert correct_counts to float and then compute the mean
            acc = torch.mean(correct_counts.type(torch.FloatTensor))
            # Compute total accuracy in the whole batch and add to train_acc
            train_acc += acc.item() * inputs.size(0)
            #print("Batch number: {:03d}, Training: Loss: {:.4f}, Accuracy:
{:.4f}".format(i, loss.item(), acc.item()))
        # Validation - No gradient tracking needed
        with torch.no_grad():
            # Set to evaluation mode
            model.eval()
            # Validation loop
            for j, (inputs, labels) in enumerate(valid_data_loader):
                inputs = inputs.to(device)
                labels = labels.to(device)
                # Forward pass - compute outputs on input data using the model
                outputs = model(inputs)
                # Compute loss
                loss = loss_criterion(outputs, labels)
                # Compute the total loss for the batch and add it to valid_loss
                valid_loss += loss.item() * inputs.size(0)
                # Calculate validation accuracy
                ret, predictions = torch.max(outputs.data, 1)
                correct_counts = predictions.eq(labels.data.view_as(predictions))
                # Convert correct_counts to float and then compute the mean
                acc = torch.mean(correct_counts.type(torch.FloatTensor))
                # Compute total accuracy in the whole batch and add to valid_acc
                valid_acc += acc.item() * inputs.size(0)
        if valid_loss < best_loss:
            best_loss = valid_loss
            best_epoch = epoch
        # Find average training loss and training accuracy
        avg_train_loss = train_loss/train_data_size
        avg_train_acc = train_acc/train_data_size
        # Find average training loss and training accuracy
        avg_valid_loss = valid_loss/valid_data_size
        avg_valid_acc = valid_acc/valid_data_size
        history.append([avg_train_loss, avg_valid_loss, avg_train_acc,
avg_valid_acc])
        epoch_end = time.time()
        print("Epoch: {:02d}/{:02d}, \tTraining:   Loss - {:.3f}, Accuracy -
{:.2f}%,\n\t\tValidation: Loss - {:.3f}, Accuracy - {:.2f}%, Time: {:.2f}s".
              format(epoch + 1, epochs, avg_train_loss, avg_train_acc * 100,
avg_valid_loss, avg_valid_acc * 100, epoch_end-epoch_start))
        # Save if the model has best accuracy till now
        torch.save(model, folder_dataset + '_model_'+str(epoch + 1) + '.pt')
    return model, history, best_epoch + 1
```

Результати числових розрахунків. У результаті проведеного

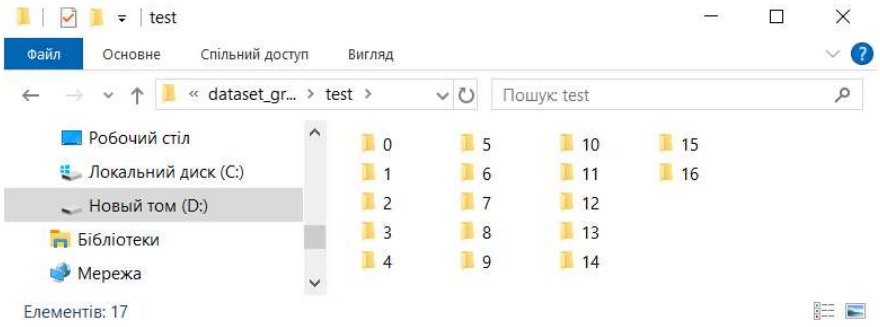
експерименту щодо тестування ефективності використання на практиці розробленого класифікатора отримано тренувальний, перевірочний та тестовий набори даних зображень оцінок, які відповідно містять 1881, 331 та 338 зображень оцінок (рис. 1, 2).

У результаті донавчання моделі ResNet50 для 50 епох отримали досить непоганий результат (рис. 3), зокрема криві навчання і перевірки досить швидко прямують до нуля (рис. 4), криві точності для навчання та перевірки дуже швидко зростають до 0,9 (рис. 5). Точність отриманої моделі стосовно розпізнавання зображень оцінок на основі згенерованого тестового набору даних становить 93,79% (рис. 6).

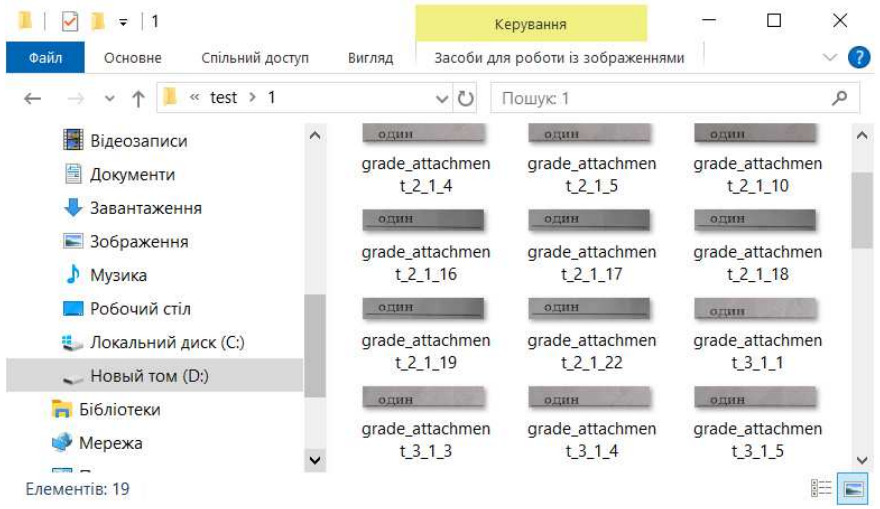
Рис. 1. Результат формування тренувальних, перевірочних та тестових наборів даних зображень оцінок



а)

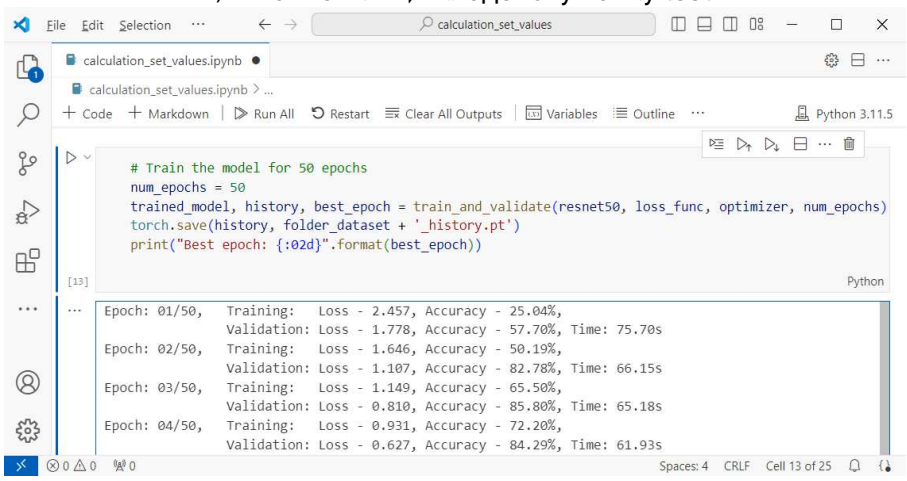


б)



в)

Рис. 2. Інформація щодо підготовки тренувальних, перевірочних та тестових наборів даних зображень оцінок: а) вміст папки dataset_grades; б) вміст папки test, вкладеної у папку dataset_grades; в) вміст папки 1, вкладеної у папку test



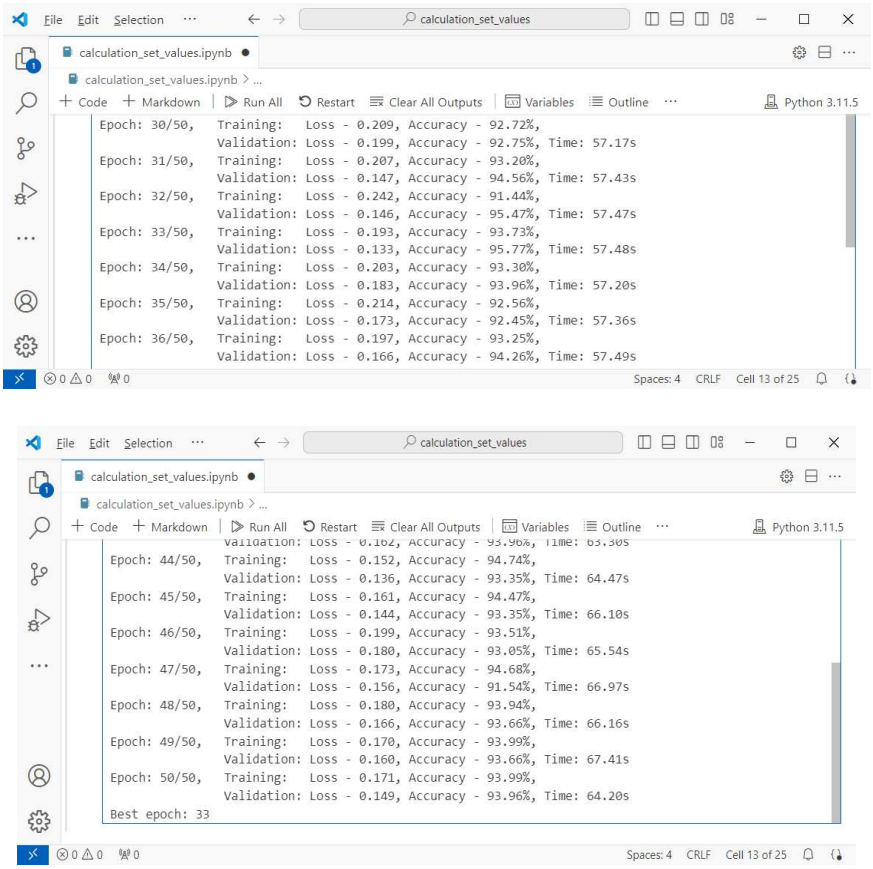


Рис. 3. Результат донавчання моделі ResNet50 для 50 епох

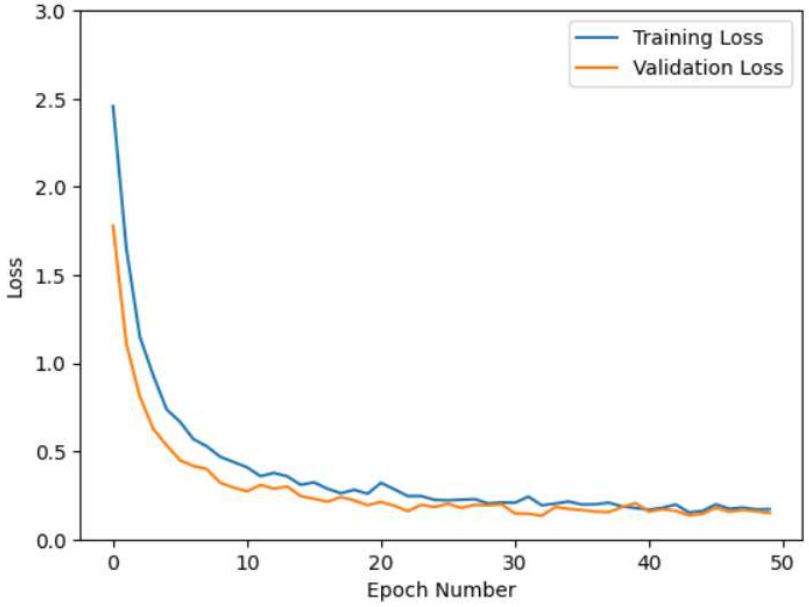


Рис. 4. Криві втрат для навчання та перевірки

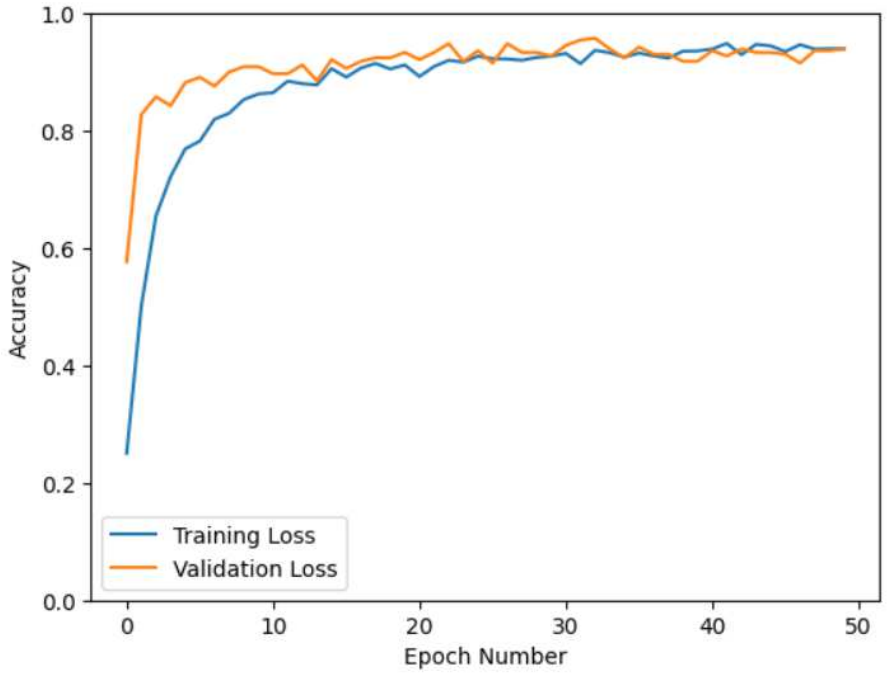


Рис. 5. Криві точності для навчання та перевірки

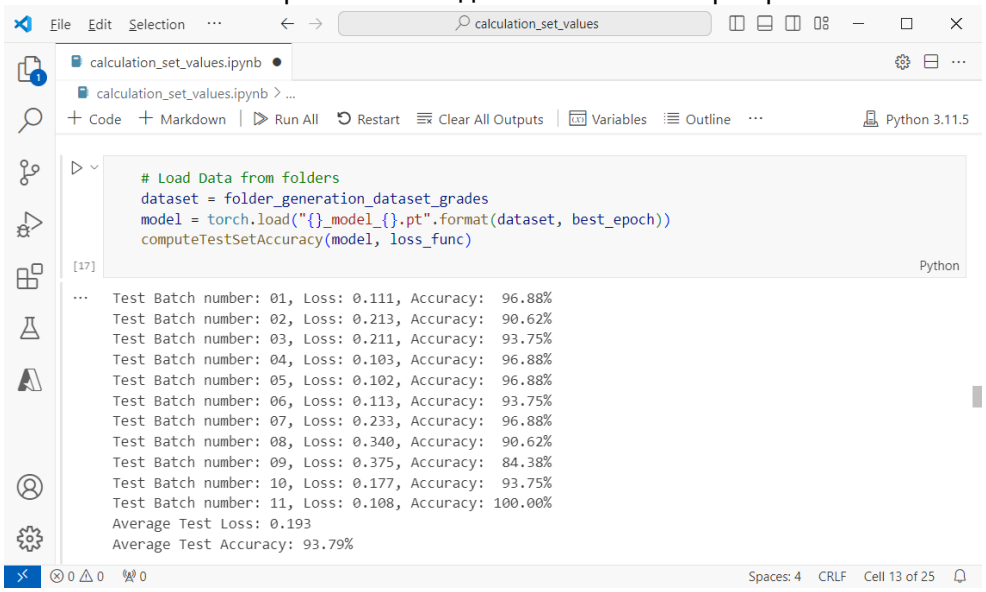


Рис. 6. Точність отриманої моделі стосовно розпізнавання зображень оцінок на основі тестового набору даних

Висновки. Розробку класифікатора зображень оцінок для розпізнавання оцінок з додатків до свідоцтв про здобуття повної загальної середньої освіти абітурієнтами реалізовано мовою програмування Python з використанням редактора VS Code та

розширення Jupyter Notebook шляхом донавчання моделі ResNet50, попередньо навченої на наборі даних ImageNet, за допомогою бібліотеки PyTorch на основі згенерованого набору даних, що містить 1881 тренувальних, 331 перевірочних та 338 тестових зображень оцінок, а для обробки зображень використано бібліотеку OpenCV. Донавчання моделі проведено для 50 епох, найкращою виявилась 33. Точність отриманої моделі стосовно розпізнавання зображень оцінок на тестовому наборі даних склала 93,79%.

Розроблений класифікатор при його донавчанні за рахунок урізноманітнення тренувальних, перевірочних та тестових наборів зображень оцінок дозволить розробити додаток, який полегшить абітурієнтам здійснювати розрахунок середнього балу свідоцтва про здобуття повної загальної середньої освіти, а працівникам приймальної комісії перевірку правильності його розрахунку на основі наданої сканкопії (фотокопії) зображення додатку до свідоцтва.

1. Вступ на бакалавра. Розрахунок середнього бала атестата у 2022 році. Освіта. 2022. URL: https://osvita.ua/consultations/bachelor/7132/ (дата звернення: 10.05.2024). **2.** Інструкція щодо роботи з системою подання заяв в електронній формі. Вступ для здобуття вищої, фахової передвищої освіти на основі свідоцтва про повну загальну середню освіту (11 класів), диплома фахового молодшого бакалавра, молодшого бакалавра, молодшого спеціаліста (НРК5). 2023. URL: https://vstup.edbo.gov.ua/files/Instrukciya_EK_VO_ PZSO_NRK5_2023.pdf (дата звернення: 10.05.2024). **3.** James Samuel. Building an Image Classifier With Pytorch. Dev. 2020. URL: https://dev.to/abiodunjames/building-an-image-classifier-using-pytorch-46dk?comments_sort=latest. (дата звернення: 10.05.2024). **4.** Neo Benedict. Building an Image Classification Model From Scratch Using PyTorch. Medium. 2021. URL: https://medium.com/bitgrit-data-science-publication/building-an-image-classification-model-with-pytorch-from-scratch-f10452073212 (дата звернення: 10.05.2024). **5.** Pointer Ian. Programming PyTorch for Deep Learning: Creating and Deploying Deep Learning Applications. O'Reilly, 2019. 226 p. **6.** Papa Joe. PyTorch Pocket Reference: Building and Deploying Deep Learning Models. O'Reilly Media, 2021. 310 p. **7.** Chung Bryan WC. Pro Processing for Images and Computer Vision with OpenCV. Apress, 2017. 301 p. **8.** Gabriel Garrido, Joshi Prateek. OpenCV 3.x with Python By Example Second Edition: Make the most of OpenCV and Python to build applications for object recognition and augmented reality. Packt Publishing, 2018. 255 p. **9.** Rosebrock Adrian. Practical Python and OpenCV: An Introductiry, Example Driven Guide to Image Processing and Computer Vision. PyImageSearch, 2016. 166 p. **10.** Fernández Villán Alberto. Mastering OpenCV 4 with Python : A

practical guide covering topics from image processing, augmented reality to deep learning with OpenCV 4 and Python 3.7. Packt Publishing : Birmingham – Mumbai, 2019. 650 p.

REFERENCES:

1. Vstup na bakalavra. Rozrakhunok serednoho bala atestata u 2022 rotsi. Osvita. 2022. URL: https://osvita.ua/consultations/bachelor/7132/ (data zvernennia: 10.05.2024). **2.** Instruktsiia shchodo roboty z systemoiu podannia zaiav v elektronnii formi. Vstup dlia zdobuttia vyshchoi, fakhovoi peredvyshchoi osvity na osnovi svidotstva pro povnu zahalnu seredniu osvitu (11 klasiv), dyploma fakhovoho molodshoho bakalavra, molodshoho bakalavra, molodshoho spetsialista (NRK5). 2023. URL: https://vstup.edbo.gov.ua/files/Instrukciya_EK_VO_ PZSO_NRK5_2023.pdf (data zvernennia: 10.05.2024). **3.** James Samuel. Building an Image Classifier With Pytorch. Dev. 2020. URL: https://dev.to/abiodunjames/building-an-image-classifier-using-pytorch-46dk?comments_sort=latest (data zvernennia: 10.05.2024). **4.** Neo Benedict. Building an Image Classification Model From Scratch Using PyTorch. Medium. 2021. URL: https://medium.com/bitgrit-data-science-publication/building-an-image-classification-model-with-pytorch-from-scratch-f10452073212 (data zvernennia: 10.05.2024). **5.** Pointer Ian. Programming PyTorch for Deep Learning: Creating and Deploying Deep Learning Applications. OReilly, 2019. 226 p. **6.** Papa Joe. PyTorch Pocket Reference: Building and Deploying Deep Learning Models. OReilly Media, 2021. 310 p. **7.** Chung Bryan WC. Pro Processing for Images and Computer Vision with OpenCV. Apress, 2017. 301 p. **8.** Gabriel Garrido, Joshi Prateek. OpenCV 3.x with Python By Example Second Edition: Make the most of OpenCV and Python to build applications for object recognition and augmented reality. Packt Publishing, 2018. 255 p. **9.** Rosebrock Adrian. Practical Python and OpenCV: An Introductiry, Example Driven Guide to Image Processing and Computer Vision. PyImageSearch, 2016. 166 p. **10.** Fernández Villán Alberto. Mastering OpenCV 4 with Python : A practical guide covering topics from image processing, augmented reality to deep learning with OpenCV 4 and Python 3.7. Packt Publishing : Birmingham – Mumbai, 2019. 650 p.

Klymiuk Yu. Ye., Candidate of Engineering (Ph.D.), Associate Professor
(National University of Water and Environmental Engineering, Rivne, yu.ye.klymiuk@nuwm.edu.ua)

DEVELOPMENT OF A CLASSIFIER OF GRADES IMAGES FOR RECOGNITION OF GRADES FROM SCAN COPIES OF ATTACHMENTS TO CERTIFICATES OF COMPLETE GENERAL SECONDARY EDUCATION OF APPLICANTS

Each year one of the most time-consuming operations for applicants when submitting applications for admission to higher education institutions in electronic form online through electronic offices is the determination of the average score of the certificate of completion of general secondary education, and for employees of admissions commissions, when processing applications submitted by applicants, the most time-consuming operation is checking the correctness of its calculation. Therefore, the development of an application for calculating the average score of a certificate of completion of general secondary education of applicants based on a scan copy (photocopy) of the image of the attachment remains a rather urgent task. To create such an application, it is necessary to create training, verification and test data sets of grades images and develop of a classifier of grades images.

The development of a classifier of grade images for recognizing grades from attachments to certificates of completion of general secondary education was implemented in the Python programming language using the VS Code editor and the Jupyter Notebook extension by retraining the ResNet50 model, pre-trained on the ImageNet dataset, using the PyTorch library based on the generated dataset containing 1881 training, 331 validation and 338 test grade images, and the OpenCV library was used for image processing. Retraining of the model was carried out for 50 epochs, 33 turned out to be the best. Accuracy of the obtained model in relation to recognition of images of grades on the test data set was 93.79%.

The developed classifier during its additional training due to the diversification of training, verification and test sets of images of grades will allow to develop an application that will make it easier for applicants to calculate the average score of a certificate for obtaining a complete general secondary education, and for employees of the

admissions committee to check the correctness of its calculation based on the provided scan copies (photo copies) of the image of the attachment to the certificate.

Keywords: **classifier; assessment; image; model; recognition; certificate; certificate; applicant.**