ДОДАТОК А
ТЕКСТ ПРОГРАМИ

ГЮИК. 502840.006–01 12 01

ЗАТВЕРДЖЕНО
ГЮИК. 502840.006–01 12 01-ЛЗ

Дослідження інтелектуальних методів аналізу рентгенівських знімків

Текст програми

ГЮИК. 502840.006–01 12 01

АРКУШІВ 7

2021 р.

Міністерство освіти і науки України
Харківський Національний Університет Радіоелектроніки

«ЗАТВЕРДЖУЮ»
Керівник кваліфікаційної роботи
доц. Петрова Р.В.

Дослідження інтелектуальних методів аналізу рентгенівських знімків

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ

ГЮИК. 502840.006–01 12 01-ЛЗ

РОЗРОБИЛА:
ст. гр. СПРм-20-1
Каменєва К.С.

2021 р

```
# In[1]:

import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# In[2]:

for dirname, _, filenames in os.walk('/kaggle/input/aptos2019-
blindness-detection/'):
    print(dirname)

# # **Visualizing Training Data**

# In[3]:

BASE_PATH='/Users/Kseniia/university/diploma2/eff2/data'
train_dataset=pd.read_csv(os.path.join(BASE_PATH,'train_labels.c
sv'))
test_dataset=pd.read_csv(os.path.join(BASE_PATH,'test_labels.csv
'))

# In[4]:

train_dataset.head(3)

# In[5]:

test_dataset.head(3)

# In[6]:

#Plot Images
from PIL import Image
from matplotlib.pyplot import imshow
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(32, 32)) #Size of Figure
columns = 3 #Columns in fig
rows = 5 #Rows in Fig
#Total images = row* columns
for i in range(1,rows*columns+1):
    IMG_PATH=BASE_PATH+'/train/'

img=Image.open(os.path.join(IMG_PATH,train_dataset.iloc[i][0]))
```

```
    fig.add_subplot(rows, columns, i)
    plt.imshow(img)
plt.show()

# ## ****Importing and Installing Libraries****

# In[7]:

#Installing Libraries
get_ipython().system('pip install efficientnet-pytorch')
get_ipython().system('pip install torchsummary')
from torchsummary import summary

# In[8]:

import PIL
import sys
import torch
from time import time
import torchvision
from PIL import Image
import torch.nn as nn
import torch.optim as optim
from torch.utils import data
from torch.autograd import Variable
import torchvision.transforms as transforms
from efficientnet_pytorch import EfficientNet

# ## **Data-Loader**

# In[9]:

class Dataset(data.Dataset):
    def __init__(self,csv_path,images_path,transform=None):
        self.train_set=pd.read_csv(csv_path)
        self.train_path=images_path
        self.transform=transform
    def __len__(self):
        return len(self.train_set)

    def __getitem__(self,idx):
        file_name=self.train_set.iloc[idx][0]
        label=self.train_set.iloc[idx][1]
        img=Image.open(os.path.join(self.train_path,file_name))
        if self.transform is not None:
            img=self.transform(img)
        return img,label
```

```
# In[10]:

training_set_untransformed=Dataset(os.path.join(BASE_PATH,'train
_labels.csv'),os.path.join(BASE_PATH,'train/'))

# ## **Defining Transforms and Parameters for Training**

# In[11]:

#Network Params
params = {'batch_size': 32,
          'shuffle': True
         }
epochs = 11
learning_rate=1e-4

# In[12]:

transform_train                                                 =
transforms.Compose([transforms.Resize((224,224)),
transforms.RandomApply([torchvision.transforms.RandomRotation(10
),
        transforms.RandomHorizontalFlip()],0.7),
transforms.ToTensor()])

# In[13]:

new_created_images=[]
for j in range (len(training_set_untransformed)):
    if training_set_untransformed[j][1]==1:
        for k in range(8):
            transformed_image                                   =
transform_train(training_set_untransformed[j][0])
            new_created_images.append((transformed_image,1))
    else:
        transformed_image                                       =
transform_train(training_set_untransformed[j][0])
        new_created_images.append((transformed_image,0))

print(len(new_created_images))

# In[14]:

train_size = int(0.8 * len(new_created_images))
validation_size = len(new_created_images) - train_size
```

```
train_dataset,                validation_dataset                =
torch.utils.data.random_split(new_created_images,
[train_size,validation_size])

# In[15]:

training_generator = data.DataLoader(train_dataset, shuffle=True,
batch_size=32, pin_memory=True)

# In[16]:

use_cuda = torch.cuda.is_available()
device = torch.device("cuda:0" if use_cuda else "cpu")
print(device)

# ## **Importing the Model (Efficient Net)**

# In[17]:

model      =      EfficientNet.from_pretrained('efficientnet-b3',
num_classes=2)

# In[18]:

model.to(device)

# In[19]:
print(summary(model, input_size=(3, 512, 512)))
# In[20]:

PATH_SAVE='./Weights/'
if(not os.path.exists(PATH_SAVE)):
    os.mkdir(PATH_SAVE)

# In[21]:

#model.load_state_dict(torch.load(os.path.join(PATH_SAVE,'4_epoc
h0.9949054896919488.pth')))

# In[22]:

criterion = nn.CrossEntropyLoss()
lr_decay=0.99
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

# # **Training The Model**
```

```
# In[23]:

eye = torch.eye(2).to(device)
classes=[0,1]

# In[24]:

history_accuracy=[]
history_loss=[]

# In[25]:

for epoch in range(epochs):
    running_loss = 0.0
    correct=0
    total=0
    class_correct = list(0. for _ in classes)
    class_total = list(0. for _ in classes)
    for i, data in enumerate(training_generator, 0):
        inputs, labels = data
        t0 = time()
        inputs, labels = inputs.to(device), labels.to(device)
        labels = eye[labels]
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, torch.max(labels, 1)[1])
        _, predicted = torch.max(outputs, 1)
        _, labels = torch.max(labels, 1)
        c = (predicted == labels.data).squeeze()
        correct += (predicted == labels).sum().item()
        total += labels.size(0)
        accuracy = float(correct) / float(total)

        history_accuracy.append(accuracy)
        history_loss.append(loss)

        loss.backward()
        optimizer.step()

        for j in range(labels.size(0)):
            label = labels[j]
            class_correct[label] += c[j].item()
            class_total[label] += 1

        running_loss += loss.item()
```

```
        print( "Epoch  :  ",epoch+1," Batch  :  ", i+1," Loss  :
",running_loss/(i+1)," Accuracy  :  ",accuracy,"Time
",round(time()-t0, 2),"s" )
    for k in range(len(classes)):
        if(class_total[k]!=0):
            print('Accuracy of %5s : %2d %%' % (classes[k], 100 *
class_correct[k] / class_total[k]))

    print('[%d epoch] Accuracy of the network on the Training
images: %d %%' % (epoch+1, 100 * correct / total))

    if epoch%10==0 or epoch==0 or epoch%1==0:
        torch.save(model.state_dict(),
os.path.join(PATH_SAVE,str(epoch+1)+'_'+str(accuracy)+'.pth'))

torch.save(model.state_dict(),
os.path.join(PATH_SAVE,'Last_epoch'+str(accuracy)+'.pth'))

# ## **Visualizing The Training Accuracy and losses**

# In[26]:

plt.plot(history_accuracy)
plt.plot(history_loss)

# # **Inference**

# In[27]:

model.load_state_dict(torch.load(os.path.join(PATH_SAVE,'Last_ep
och0.9973794549266247.pth')))

# In[28]:

model.eval()

# In[29]:

test_transforms                                                =
transforms.Compose([transforms.Resize((224,224)),
transforms.RandomApply([torchvision.transforms.RandomRotation(10
),                      transforms.RandomHorizontalFlip()],0.7),
transforms.ToTensor(),])

# In[30]:

def predict_image(image):
```

```
    image_tensor = test_transforms(image)
    image_tensor = image_tensor.unsqueeze_(0)
    input = Variable(image_tensor)
    input = input.to(device)
    output = model(input)
    index = output.data.cpu().numpy().argmax()
    return index

# In[31]:

correct_counter=0
for i in range(len(validation_dataset)):
    image_tensor = validation_dataset[i][0].unsqueeze_(0)
    input = Variable(image_tensor)
    input = input.to(device)
    output = model(input)
    index = output.data.cpu().numpy().argmax()
    if index == validation_dataset[i][1]:
        correct_counter+=1
print("Accuracy=",correct_counter/len(validation_dataset))

# In[32]:

submission=pd.read_csv(BASE_PATH+'/sample_submission.csv')

# In[33]:

submission.head(3)

# In[34]:

submission_csv=pd.DataFrame(columns=['id_code','diagnosis'])

# In[35]:

IMG_TEST_PATH=os.path.join(BASE_PATH,'test/')
for i in range(len(submission)):
    img=Image.open(IMG_TEST_PATH+submission.iloc[i][0])
    prediction=predict_image(img)
    submission_csv=submission_csv.append({'id_code':
submission.iloc[i][0],'diagnosis': prediction},ignore_index=True)
    if(i%10==0 or i==len(submission)-1):
        print('[',32*'=','>]
',round((i+1)*100/len(submission),2),' % Complete')
# In[36]:

submission_csv.to_csv('submission.csv',index=False)
```

ДОДАТОК Б
ГРАФІЧНИЙ МАТЕРІАЛ КВАЛІФІКАЦІЙНОЇ РОБОТИ

ГЮИК. 502840.006

ЗАТВЕРДЖЕНО
ГЮИК. 502840.006 – ЛЗ

Дослідження інтелектуальних методів аналізу рентгенівських знімків

Графічний матеріал

ГЮИК. 502840.006 – ЛЗ

АРКУШІВ 5

2021 р.

Міністерство освіти і науки України
Харківський Національний Університет Радіоелектроніки

«ЗАТВЕРДЖУЮ»
Керівник кваліфікаційної роботи
доц. Петрова Р.В.

Дослідження інтелектуальних методів аналізу рентгенівських знімків

Графічний матеріал

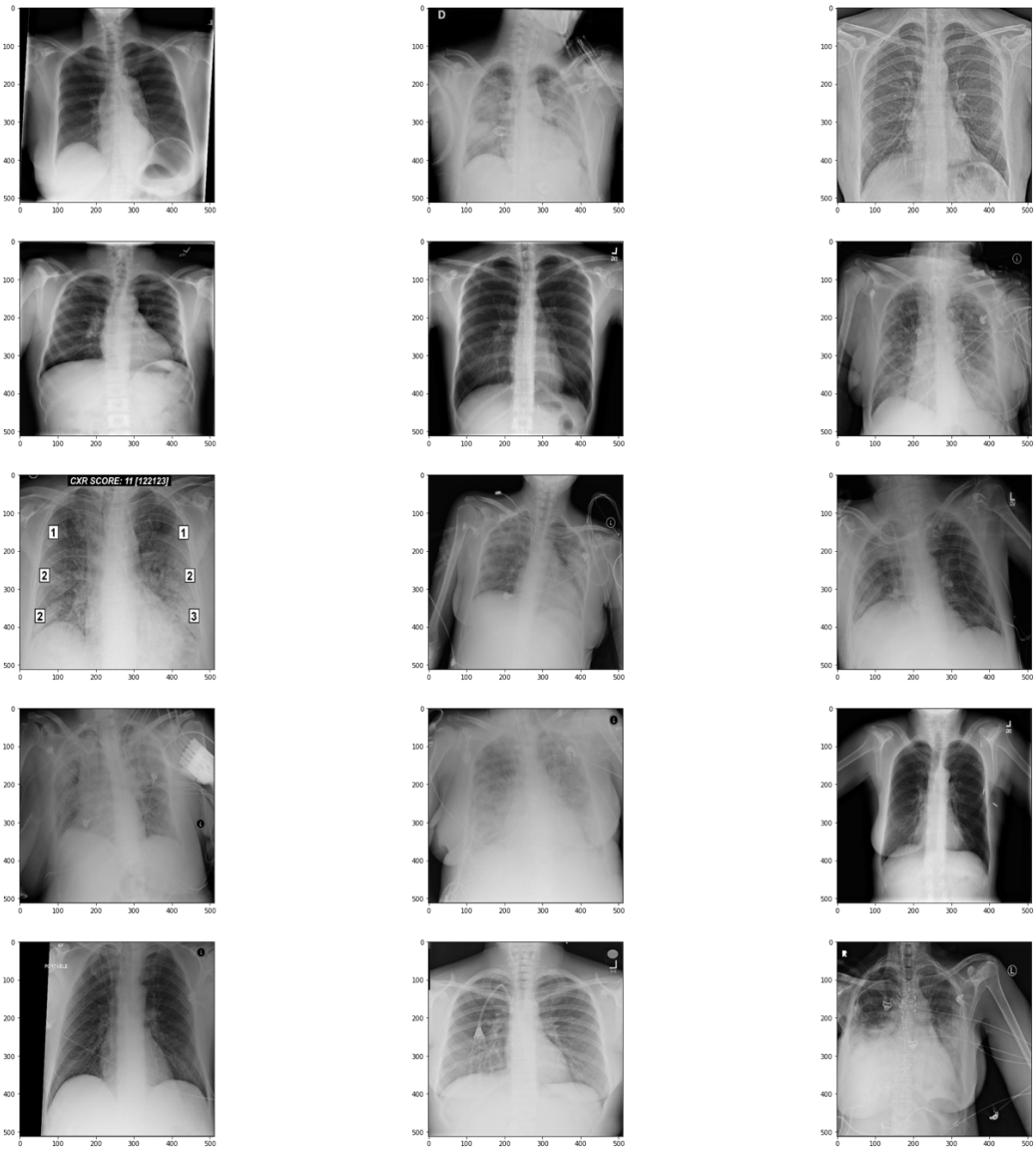
ЛИСТ ЗАТВЕРДЖЕННЯ

ГЮИК. 502840.006 – ЛЗ

РОЗРОБИЛА:
ст. гр. СПРм-20-1
Каменєва К.С.

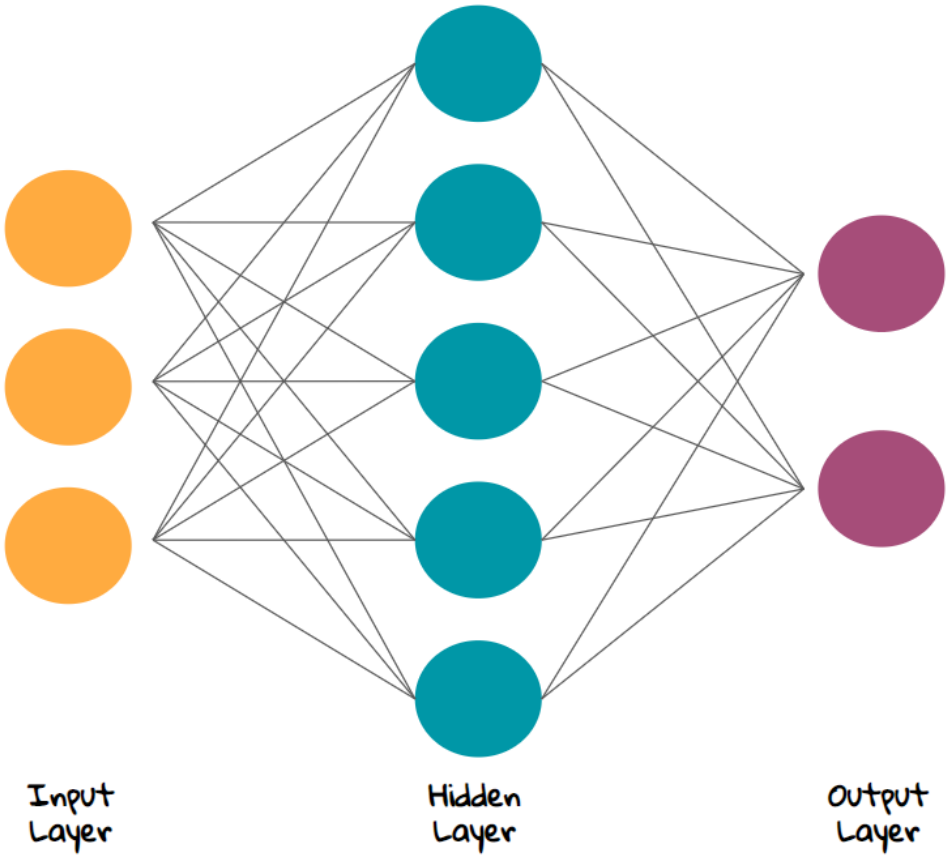
2021 р.

ВІДОБРАЖЕННЯ ЗОБРАЖЕНЬ РЕНТГЕНІВСЬКИХ ЗНІМКІВ ЛЕГЕНІВ З НАВЧАЛЬНОГО НАБОРУ



Розробив	Каменєва К.С.			Дослідження інтелектуальних методів аналізу рентгенівських знімків	
Перевірила	доц. Петрова Р.В.				
Н. Контр.	доц. Петрова Р.В.				
				СПРм-20-1	Аркуш 1
Затвердив	Гребеннік І.В.			СТ	Аркушів 1

СТРУКТУРА ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ

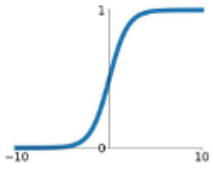


Розробив	Каменєва К.С.			Дослідження інтелектуальних методів аналізу рентгенівських знімків	
Перевірила	доц. Петрова Р.В.				
Н. Контр.	доц. Петрова Р.В.				
				СПРм-20-1	Аркуш 1
Затвердив	Гребеннік І.В.			СТ	Аркушів 1

ФУНКЦІЇ АКТИВАЦІЇ ТА ЇХ ГРАФІКИ

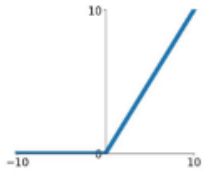
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



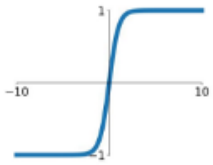
ReLU

$$\max(0, x)$$



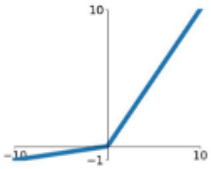
tanh

$$\tanh(x)$$



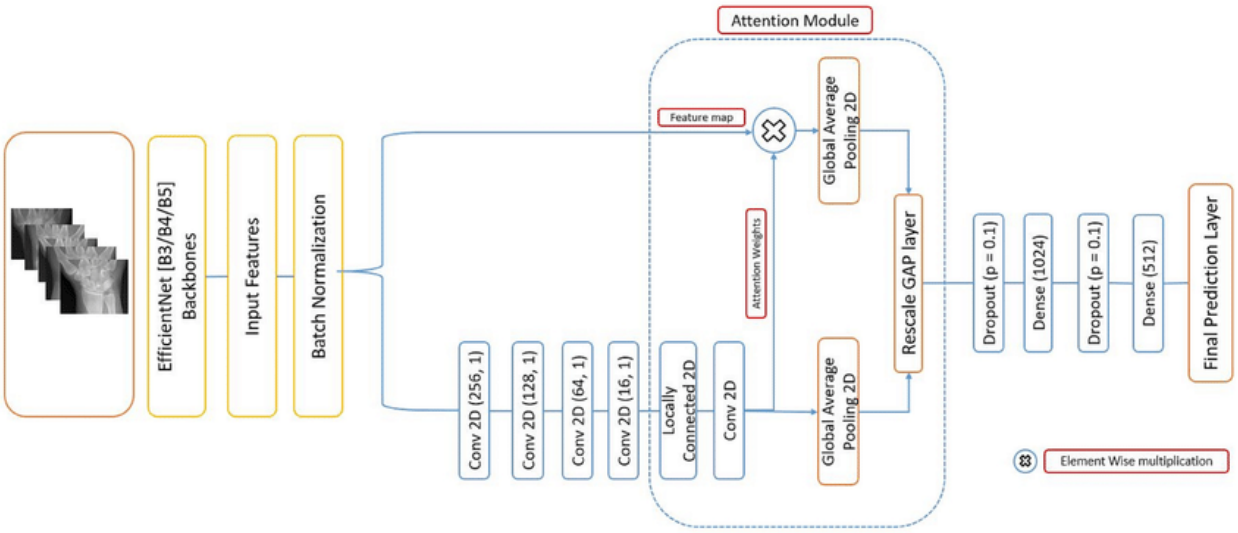
Leaky ReLU

$$\max(0.1x, x)$$



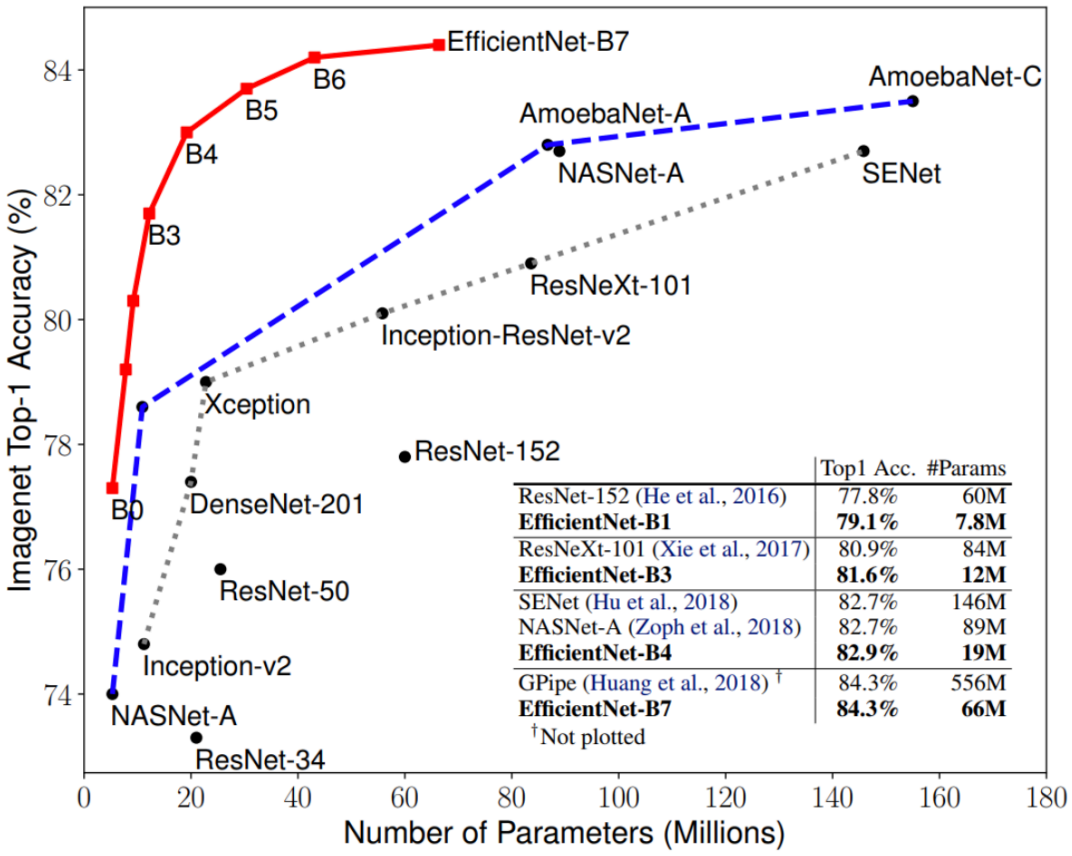
Розробив	Каменєва К.С.			Дослідження інтелектуальних методів аналізу рентгенівських знімків	
Перевірила	доц. Петрова Р.В.				
Н. Контр.	доц. Петрова Р.В.				
				СПРм-20-1	Аркуш 1
Затвердив	Гребеннік І.В.			СТ	Аркушів 1

АРХІТЕКТУРА НЕЙРОННОЇ МЕРЕЖІ EFFICIENTNET



Розробив	Каменєва К.С.			Дослідження інтелектуальних методів аналізу рентгенівських знімків	
Перевірила	доц. Петрова Р.В.				
Н. Контр.	доц. Петрова Р.В.				
				СПРм-20-1	Аркуш 1
Затвердив	Гребеннік І.В.			СТ	Аркушів 1

ПРОДУКТИВНІСТЬ EFFICIENTNET ДЛЯ НАБОРУ ДАНИХ IMAGENET В ПОРІВНЯННІ З ІНШИМИ АРХІТЕКТУРАМИ МЕРЕЖ



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Розробив	Каменєва К.С.			Дослідження інтелектуальних методів аналізу рентгенівських знімків	
Перевірила	доц. Петрова Р.В.				
Н. Контр.	доц. Петрова Р.В.				
				СПРм-20-1	Аркуш 1
Затвердив	Гребеннік І.В.			СТ	Аркушів 1

ДОДАТОК В
ВІДОМІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ МАГІСТРА

ГЮИК. 502840.006 ДЗ

(позначення документу)

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1.	ГЮИК. 502840.006 ПЗ	Пояснювальна записка	98 стор.
2.	ГЮИК. 502840.006 -01 12 01	Текст програми	7 стор.
		Графічні документи	
4.		Відображення зображень рентгенівських знімків легенів	1 аркуш
		з навчального набору	
5.		Структура штучної нейронної мережі	1 аркуш
6.		Функції активації та їх графіки	1 аркуш
7.		Архітектура нейронної мережі Efficientnet	1 аркуш
8.		Продуктивність Efficientnet для набору даних Imagenet в	1 аркуш
		порівнянні з іншими архітектурами мереж	

					ГЮИК. 502840.006 Д3	
.						
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Каменєва К.С.*			*«Дослідження інтелектуальних методів аналізу рентгенівських знімків»*	Аркуш: *1*; Аркушів: *1*
Перевірив.		*доц. Петрова Р.В.*				
Н. контроль		*доц. Петрова Р.В.*				*ХНУРЕ Кафедра СТ*
Затвердив		*Гребеннік І.В.*				