MODEL PENGEMBANGAN CONVOLUTIONAL NEURAL NETWORK UNTUK MENGKLASIFIKASI PNEUMONIA DENGAN PENINGKATAN VARIASI CITRA TORAKS MENGGUNAKAN AUGMENTASI CITRA

A. Deskripsi singkat

Merupakan bagian dari penelitian yang bertopik di pneumonia. Model ini bertujuan untuk meningkatkan variasi citra toraks yang mana dengan semakin banyaknya variasi citra diharapkan akan meningkatkan akurasi klasifikasi dari model CNN.

B. Script penerapan Albumentasi dalam peningkatan variasi citra toraks

```
!pip uninstall opencv-python-headless
!pip install opencv-python-headless==4.1.2.30

%pip install -U albumentations

import os,glob,random,shutil,time,pathlib
from tqdm import tqdm

import cv2
from matplotlib import pyplot as plt

import albumentations as A

import numpy as np
from PIL import Image as im

count = 0
inp_pneu= "dataset/pneumonia"
out_backup_pneu="dataset-backup/PNEUMONIA"
out_train_pneu= "dataset-norm/TRAINING/PNEUMONIA"
out_vali_pneu= "dataset-norm/VALIDATION/PNEUMONIA"
out_test_pneu= "dataset-norm/TESTING/PNEUMONIA"
# %cd content/drive/MyDrive/PNEUMONIA/albumentations
# %ls
pwd
def visualize(image):
    plt.figure(figsize=(10, 10))
    plt.axis('off')
    plt.imshow(image)

image =
cv2.imread('/content/drive/MyDrive/PNEUMONIA/albumentations/download.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
visualize(image)

transform = A.Compose([
    A.CLAHE(),
    A.RandomRotate90(),
    A.Transpose(),
    A.ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.50, rotate_limit=45,
p=.75),
    A.Blur(blur_limit=3),
    A.OpticalDistortion(),
    A.GridDistortion(),
    A.HueSaturationValue(),
])
random.seed(42)
augmented_image = transform(image=image)['image']
visualize(augmented_image)
%pwd

count = 0
imginp= "/content/drive/MyDrive/PNEUMONIA/albumentations"
imgout= "/content/drive/MyDrive/PNEUMONIA/albumentations/output"

if not os.path.isdir(imgout):
    os.makedirs(imgout)

for path in os.listdir(imginp):
  if os.path.isfile(os.path.join(imginp, path)):
      count += 1
print("Number of Files in directory: " ,count)
# print("Moving Files to Training. . .")
start = time.time()

for root, dirs, files in os.walk(os.path.abspath(imginp)):
    for file in files:
        print(os.path.join(root, file))
        cit=os.path.join(root, file)

# end = time.time()
# print("Process is finished in: ",end - start, " second\n")
%pwd
listoff= [f for f in os.listdir(imginp) if os.path.isfile(os.path.join(imginp,
f))]
print(listoff)
# Get the list of all files and directories
path = "/content/drive/MyDrive/PNEUMONIA/albumentations"
dir_list = os.listdir(path)

print("Files and directories in '", path, "' :")
```

```
for x in dir_list:
    if x.endswith(".jpg"):
        # Prints only text file present in My Folder
        print(x)

# prints all files
# print(dir_list)
from PIL import Image
import PIL

# making a function to load the image
def load_img(path):
    image = cv2.imread(path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return image

# load the image
# image = load_img(cit)
# image.shape

# Augmentation Pipeline with Albumentations
transform = A.Compose([
    A.Blur(blur_limit=(60, 70), p=0.5),
    A.Resize(512, 512),
    A.RandomBrightnessContrast(p=0.5),
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.5)
])

# transformed_img = transform(image=image)['image']
# transformed_img.shape

for root, dirs, files in os.walk(os.path.abspath(imginp)):
    for file in files:
      for x in dir_list:
        if x.endswith(".jpg"):

          print(os.path.join(root, file))
          cit=os.path.join(root, file)

          image = load_img(cit)
          # image.shape

          transformed_img = transform(image=image)['image']

          data = im.fromarray(transformed_img)
```

```
          # data.save("output/"+'zzz.jpg')

          print(file)
          print(x)

        # comparison(image, transformed_img)

#============================
# Importing Image module from PIL package


# creating a image object (main image)
# im1 = Image.open(transformed_img)

# save a image using extension
# im1 =
im1.save("/content/drive/MyDrive/PNEUMONIA/albumentations/output/geeks.jpg")
# print(type(im1))
print(type(transformed_img))
print(type(cit))
print(type(image))

print(transformed_img.shape)
import numpy as np
from PIL import Image as im


# this function takes two image and plots it side-by-side for comparison
# [OPTIONAL: its fine if you don't see how this works as it has nothing to do
with data augmentation]
def comparison(actual_image, transformed_image, titles=None):
    if titles == None:
        titles = ['Actual Image', 'Transformed Image']

    fig, ax = plt.subplots(1, 2, figsize=(17, 17))
    ax[0].set_title(titles[0])
    if len(actual_image.shape) == 2:
        ax[0].imshow(actual_image, cmap='gray')
    else:
        ax[0].matshow(actual_image)
    ax[0].axis('off')

    ax[1].set_title(titles[1])
    if len(transformed_image.shape) == 2:
        ax[1].imshow(transformed_image, cmap='gray')
    else:
        ax[1].matshow(transformed_image)
    ax[1].axis('off')
    plt.show()
```

```
# comparison(image, transformed_img)

# define a main function
def main():

    # create a numpy array from scratch
    # using arange function.
    # 1024x720 = 737280 is the amount
    # of pixels.
    # np.uint8 is a data type containing
    # numbers ranging from 0 to 255
    # and no non-negative integers
    array = np.arange(0, 737280, 1, np.uint8)

    # check type of array
    print(type(array))

    # our array will be of width
    # 737280 pixels That means it
    # will be a long dark line
    print(array.shape)

    # Reshape the array into a
    # familiar resoluition
    array = np.reshape(array, (1024, 720))

    # show the shape of the array
    print(array.shape)

    # show the array
    print(array)

    # creating image object of
    # above array
    data = im.fromarray(transformed_img)

    # saving the final output
    # as a PNG file
    data.save('gfg_dummy_pic2.png')

# driver code
if __name__ == "__main__":

  # function call
  main()
pwd
for root, dirs, files in os.walk(".", topdown=False):
```

```
  #  for name in files:
      # print(os.path.join(root, name))
   for name in dirs:
      print(os.path.join(root, name))
image = PIL.Image.fromarray(transformed_img, "RGB")
```

C. Script penerapan CNN dalam klasifikasi

```
import os,glob,random,shutil,time
from tqdm import tqdm

count = 0
inp_pneu= "dataset/pneumonia"
out_backup_pneu="dataset-backup/PNEUMONIA"
out_train_pneu= "dataset-norm/TRAINING/PNEUMONIA"
out_vali_pneu= "dataset-norm/VALIDATION/PNEUMONIA"
out_test_pneu= "dataset-norm/TESTING/PNEUMONIA"

if not os.path.isdir(out_backup_pneu):
    os.makedirs(out_backup_pneu)
if not os.path.isdir(out_train_pneu):
    os.makedirs(out_train_pneu)
if not os.path.isdir(out_vali_pneu):
    os.makedirs(out_vali_pneu)
if not os.path.isdir(out_test_pneu):
    os.makedirs(out_test_pneu)


# ======================move to training folder
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files in Dataset Pneumonia: " ,count)
print("Moving Files to Training. . .")
start = time.time()
for i in tqdm(range(int(0.8*count))):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_train_pneu+"/"+
random_img)
```

```
end = time.time()
print("Process is finished in: ",end - start, " second\n")

# ======================move to validation folder
count=0
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files LEFT in Dataset Pneumonia: " ,count)
print("Moving Files to Validation. . .")
start = time.time()
for i in tqdm(range(int(0.5*count))):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_vali_pneu+"/"+r
andom_img)
end = time.time()
print("Process is finished in: ",end - start, " second\n")

# ==================move to testing folder
count=0
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files LEFT in Dataset Pneumonia: " ,count)
print("Moving Files to Testing. . .")
start = time.time()
for i in tqdm(range(count)):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_test_pneu+"/"+r
andom_img)
end = time.time()
print("Process is finished in: ",end - start, " second")


import os,glob,random,shutil,time
```

```
from tqdm import tqdm

count = 0
inp_pneu= "dataset/NORMAL"
out_backup_pneu="dataset-backup/NORMAL"
out_train_pneu= "dataset-norm/TRAINING/NORMAL"
out_vali_pneu= "dataset-norm/VALIDATION/NORMAL"
out_test_pneu= "dataset-norm/TESTING/NORMAL"

if not os.path.isdir(out_backup_pneu):
    os.makedirs(out_backup_pneu)
if not os.path.isdir(out_train_pneu):
    os.makedirs(out_train_pneu)
if not os.path.isdir(out_vali_pneu):
    os.makedirs(out_vali_pneu)
if not os.path.isdir(out_test_pneu):
    os.makedirs(out_test_pneu)


# ======================move to training folder
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files in Dataset NORMAL: " ,count)
print("Moving Files to Training. . .")
start = time.time()
for i in tqdm(range(int(0.8*count))):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_train_pneu+"/"+
random_img)

end = time.time()
print("Process is finished in: ",end - start, " second \n")

# ======================move to validation folder
count=0
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files LEFT in Dataset Normal: " ,count)
print("Moving Files to Validation. . .")
start = time.time()
```

```
for i in tqdm(range(int(0.5*count))):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_vali_pneu+"/"+r
andom_img)
end = time.time()
print("Process is finished in: ",end - start, " second\n")

# ==================move to testing folder
count=0
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files LEFT in Dataset NORMAL: " ,count)
print("Moving Files to Testing. . .")
start = time.time()
for i in tqdm(range(count)):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_test_pneu+"/"+r
andom_img)
end = time.time()
print("Process is finished in: ",end - start, " second")
#copy from dataset-backup


import os,glob,random,shutil,time
from tqdm import tqdm

count=0
count2 = 0
source= "dataset-backup/pneumonia"
source2= "dataset-backup/normal"
dest="dataset/PNEUMONIA"
dest2="dataset/NORMAL"
delit="dataset-norm"
delit2="dataset-backup"
```

```
if not os.path.isdir(dest):
    os.makedirs(dest)

for path in os.listdir(source):
    if os.path.isfile(os.path.join(source, path)):
        count += 1
print("Number of Files in Dataset-Backup Pneumonia: " ,count)
print("Moving Files to Original Location")
for path in os.listdir(source2):
    if os.path.isfile(os.path.join(source2, path)):
        count2 += 1
print("Number of Files in Dataset-Backup NORMAL: " ,count2)
print("Moving Files to Original Location")

if count!=0 or count2!=0:
    start = time.time()

    for i in tqdm(range(count)):
        listoff= [f for f in os.listdir(source) if os.path.isfile(os.pa
th.join(source, f))]
        random_img = random.choice(listoff)
    #     print(i+1,"isi random_img: ",random_img)
        shutil.move(source+"/"+random_img,dest+"/"+random_img)
    for i in tqdm(range(count2)):
        listoff2= [f for f in os.listdir(source2) if os.path.isfile(os.
path.join(source2, f))]
        random_img = random.choice(listoff2)
    #     print(i+1,"isi random_img: ",random_img)
        shutil.move(source2+"/"+random_img,dest2+"/"+random_img)

shutil.rmtree(delit)
shutil.rmtree(delit2)

end = time.time()
print("Process is finished in: ",end - start, " second")
#split bacteria

import os,glob,random,shutil,time
from tqdm import tqdm

count = 0
inp_pneu= "dataset/bacteria"
out_backup_pneu="dataset-backup/bacteria"
out_train_pneu= "dataset-bv/TRAINING/bacteria"
out_vali_pneu= "dataset-bv/VALIDATION/bacteria"
out_test_pneu= "dataset-bv/TESTING/bacteria"
```

```
if not os.path.isdir(out_backup_pneu):
    os.makedirs(out_backup_pneu)
if not os.path.isdir(out_train_pneu):
    os.makedirs(out_train_pneu)
if not os.path.isdir(out_vali_pneu):
    os.makedirs(out_vali_pneu)
if not os.path.isdir(out_test_pneu):
    os.makedirs(out_test_pneu)


# ======================move to training folder
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files in Dataset Bacteria: " ,count)
print("Moving Files to Training. . .")
start = time.time()
for i in tqdm(range(int(0.8*count))):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_train_pneu+"/"+
random_img)

end = time.time()
print("Process is finished in: ",end - start, " second\n")

# ======================move to validation folder
count=0
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files LEFT in Dataset Bacteria: " ,count)
print("Moving Files to Validation. . .")
start = time.time()
for i in tqdm(range(int(0.5*count))):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
```

```
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_vali_pneu+"/"+r
andom_img)
end = time.time()
print("Process is finished in: ",end - start, " second\n")

# ==================move to testing folder
count=0
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files LEFT in Dataset Bacteria: " ,count)
print("Moving Files to Testing. . .")
start = time.time()
for i in tqdm(range(count)):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_test_pneu+"/"+r
andom_img)
end = time.time()
print("Process is finished in: ",end - start, " second")


#split virus

import os,glob,random,shutil,time
from tqdm import tqdm

count = 0
inp_pneu= "dataset/Virus"
out_backup_pneu="dataset-backup/Virus"
out_train_pneu= "dataset-bv/TRAINING/Virus"
out_vali_pneu= "dataset-bv/VALIDATION/Virus"
out_test_pneu= "dataset-bv/TESTING/Virus"

if not os.path.isdir(out_backup_pneu):
    os.makedirs(out_backup_pneu)
if not os.path.isdir(out_train_pneu):
    os.makedirs(out_train_pneu)
if not os.path.isdir(out_vali_pneu):
    os.makedirs(out_vali_pneu)
if not os.path.isdir(out_test_pneu):
    os.makedirs(out_test_pneu)
```

```
# ======================move to training folder
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files in Dataset Virus: " ,count)
print("Moving Files to Training. . .")
start = time.time()
for i in tqdm(range(int(0.8*count))):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_train_pneu+"/"+
random_img)

end = time.time()
print("Process is finished in: ",end - start, " second\n")

# ======================move to validation folder
count=0
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
        count += 1
print("Number of Files LEFT in Dataset Virus: " ,count)
print("Moving Files to Validation. . .")
start = time.time()
for i in tqdm(range(int(0.5*count))):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_vali_pneu+"/"+r
andom_img)
end = time.time()
print("Process is finished in: ",end - start, " second\n")

# ==================move to testing folder
count=0
for path in os.listdir(inp_pneu):
    if os.path.isfile(os.path.join(inp_pneu, path)):
```

```
        count += 1
print("Number of Files LEFT in Dataset Virus: " ,count)
print("Moving Files to Testing. . .")
start = time.time()
for i in tqdm(range(count)):

    listoff= [f for f in os.listdir(inp_pneu) if os.path.isfile(os.path
.join(inp_pneu, f))]
    random_img = random.choice(listoff)
#     print(i+1,"isi random_img: ",random_img)

    new_path = shutil.copy(inp_pneu+"/"+random_img,out_backup_pneu+"/"+
random_img)
    new_path2 = shutil.move(inp_pneu+"/"+random_img,out_test_pneu+"/"+r
andom_img)
end = time.time()
print("Process is finished in: ",end - start, " second")


#copy from dataset-backup to bacteria and virus


import os,glob,random,shutil,time
from tqdm import tqdm

count=0
count2 = 0
source= "dataset-backup/bacteria"
source2= "dataset-backup/virus"
dest="dataset/bacteria"
dest2="dataset/virus"
delit="dataset-bv"
delit2="dataset-backup"

if not os.path.isdir(dest):
    os.makedirs(dest)

for path in os.listdir(source):
    if os.path.isfile(os.path.join(source, path)):
        count += 1
print("Number of Files in Dataset-Backup BACTERIA: " ,count)
print("Moving Files to Original Location")
for path in os.listdir(source2):
    if os.path.isfile(os.path.join(source2, path)):
        count2 += 1
print("Number of Files in Dataset-Backup VIRUS: " ,count2)
print("Moving Files to Original Location")

if count!=0 or count2!=0:
```

```
    start = time.time()

    for i in tqdm(range(count)):
        listoff= [f for f in os.listdir(source) if os.path.isfile(os.pa
th.join(source, f))]
        random_img = random.choice(listoff)
    #     print(i+1,"isi random_img: ",random_img)
        shutil.move(source+"/"+random_img,dest+"/"+random_img)
    for i in tqdm(range(count2)):
        listoff2= [f for f in os.listdir(source2) if os.path.isfile(os.
path.join(source2, f))]
        random_img = random.choice(listoff2)
    #     print(i+1,"isi random_img: ",random_img)
        shutil.move(source2+"/"+random_img,dest2+"/"+random_img)

shutil.rmtree(delit)
# shutil.rmtree(delit2)

end = time.time()
print("Process is finished in: ",end - start, " second")
```