Laporan Project UAS Machine Learning

C14210005 Joshua Briantama Hanjaya
C14210177 Alfred Wisana
C14210185 Frederika Handoyo

Dataset
https://www.kaggle.com/competitions/dogs-vs-cats/data?select=train.zip
dataset ini berisi 25000 image dengan label 0 untuk cat dan 1 untuk dog
bentuk dataset ini adalah image dengan extension jpg yang di zip menjadi 2 file yaitu :
train.zip (isi nya campur cat dan dog)
test1.zip
kita diminta untuk membuat model yang dapat mengklasifikasikan sebuah gambar termasuk cat atau dog

Preprocessing
kita pisahkan data dari train.zip menjadi :
cat training
dog training
cat validation
dog validation

```
# create directories
dataset_home = 'C:/Users/LENOVO/Documents/school/SEM 5/Machine Learning/datasets/dataset_dogs_vs_cats/'
subdirs = ['train/', 'test/']

for subdir in subdirs:
    # create label subdirectories
    labeldirs = ['dogs/', 'cats/']
    for labldir in labeldirs:
        newdir = dataset_home + subdir + labldir
        makedirs(newdir, exist_ok=True)
# seed random number generator
seed(1)
# define ratio of pictures to use for validation
val_ratio = 0.2
# copy training dataset images into subdirectories
src_directory = 'C:/Users/LENOVO/Documents/school/SEM 5/Machine Learning/datasets/train/'
for file in listdir(src_directory):
        src = src_directory + '/' + file
        dst_dir = 'train/'
        if random() < val_ratio:
            dst_dir = 'test/'
        if file.startswith('cat'):
            dst = dataset_home + dst_dir + 'cats/' + file
            copyfile(src, dst)
        elif file.startswith('dog'):
            dst = dataset_home + dst_dir + 'dogs/' + file
            copyfile(src, dst)


path1 = "C:/Users/LENOVO/Documents/school/SEM 5/Machine Learning/datasets/dataset_dogs_vs_cats/train/cats"
path2 = "C:/Users/LENOVO/Documents/school/SEM 5/Machine Learning/datasets/dataset_dogs_vs_cats/train/dogs"
path3 = "C:/Users/LENOVO/Documents/school/SEM 5/Machine Learning/datasets/dataset_dogs_vs_cats/test/cats"
path4 = "C:/Users/LENOVO/Documents/school/SEM 5/Machine Learning/datasets/dataset_dogs_vs_cats/test/dogs"

print('Then number of cat images in training data is' ,len(os.listdir(path1)))
print('Then number of dog images in training data is' ,len(os.listdir(path2)))
print('Then number of cat images in validation data is' ,len(os.listdir(path3)))
print('Then number of dog images in validation data is' ,len(os.listdir(path4)))
```

```
Then number of cat images in training data is 9945
Then number of dog images in training data is 9965
Then number of cat images in validation data is 2555
Then number of dog images in validation data is 2535
```

Setelah kita pisah kedalam suatu folder maka dapat terlihat distribsusi data cat dan dog pada testing sama dan juga demikian pada validation (kurang lebih sama)

Selanjutnya kita akan memasukkan semua kedalam array training dan array test(validation)

```
img_size = 100
train_data = []
test_data = []
```

```
count = 0
for img in os.listdir(DIRECTORY_train_cats):
    img_path = os.path.join(DIRECTORY_train_cats, img)
    img_arr = cv2.imread(img_path)
    img_arr = cv2.resize(img_arr, (img_size, img_size))
    train_data.append([img_arr, 0])
    count+=1
print('jumlah train cat = ',count)
```

```
jumlah train cat =  9945
```

```
count = 0
for img in os.listdir(DIRECTORY_train_dogs):
    img_path = os.path.join(DIRECTORY_train_dogs, img)
    img_arr = cv2.imread(img_path)
    img_arr = cv2.resize(img_arr, (img_size, img_size))
    train_data.append([img_arr, 1])
    count+=1
print('jumlah train dog = ',count)
```

```
jumlah train dog =  9965
```

```
count = 0
for img in os.listdir(DIRECTORY_test_cats):
    img_path = os.path.join(DIRECTORY_test_cats, img)
    img_arr = cv2.imread(img_path)
    img_arr = cv2.resize(img_arr, (img_size, img_size))
    test_data.append([img_arr, 0])
    count+=1
print('jumlah test cat = ',count)
```

```
jumlah test cat =  2555
```

```
count = 0
for img in os.listdir(DIRECTORY_test_dogs):
    img_path = os.path.join(DIRECTORY_test_dogs, img)
    img_arr = cv2.imread(img_path)
    img_arr = cv2.resize(img_arr, (img_size, img_size))
    test_data.append([img_arr, 1])
    count+=1
print('jumlah test dog = ',count)
```

```
jumlah test dog =  2535
```

shuffle array train dan test data, masukkan data ke array
X_train, y_train, X_test, y_test

save setiap array tersebut menggunakan pickle

```
print(len(train_data))
print(len(test_data))
```

```
19910
5090
```

```
random.shuffle(train_data)
random.shuffle(test_data)
```

```
X_train = []
y_train = []
X_test = []
y_test = []

for features, labels in train_data:
    X_train.append(features)
    y_train.append(labels)

for features, labels in test_data:
    X_test.append(features)
    y_test.append(labels)
```

```
len(X_train)
```

```
19910
```

```
len(X_test)
```

```
5090
```

```
X_train = np.array(X_train)
y_train = np.array(y_train)
X_test = np.array(X_test)
y_test = np.array(y_test)
```

```
pickle.dump(X_train, open('X_train.pkl','wb'))
pickle.dump(y_train, open('y_train.pkl','wb'))
pickle.dump(X_test, open('X_test.pkl','wb'))
pickle.dump(y_test, open('y_test.pkl','wb'))
```

Pada saat kita ingin memakai dataset tersebut kita akan open dan normalisasi seperti biasa

```
import pickle
```

```
X_train = pickle.load(open('X_train.pkl', 'rb'))
y_train = pickle.load(open('y_train.pkl', 'rb'))
X_test = pickle.load(open('X_test.pkl', 'rb'))
y_test = pickle.load(open('y_test.pkl', 'rb'))
```

```
X_train = X_train/255
X_test = X_test/255
```

```
print(X_train.shape)
print(X_test.shape)
```

```
(19910, 100, 100, 3)
(5090, 100, 100, 3)
```

Jika kita menggunakan augmentasi data

```
# Creating image data generator
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range = 15,
                                   horizontal_flip = True,
                                   zoom_range = 0.2,
                                   shear_range = 0.1,
                                   fill_mode = 'reflect',
                                   width_shift_range = 0.1,
                                   height_shift_range = 0.1)

test_datagen = ImageDataGenerator(rescale=1./255)
```

```
train_gen = train_datagen.flow_from_directory('C:/Users/LENOVO/Documents/school/SEM 5/Machine Learning/dat
                                              class_mode='binary',
                                              target_size = (image_size,image_size),
                                              batch_size = bat_size,
                                             )

val_gen = test_datagen.flow_from_directory('C:/Users/LENOVO/Documents/school/SEM 5/Machine Learning/datase
                                           class_mode='binary',
                                           batch_size = bat_size,
                                           target_size = (image_size,image_size),
                                           shuffle = False
                                          )
```

```
Found 19910 images belonging to 2 classes.
Found 5090 images belonging to 2 classes.
```

SEMUA DIJALANKAN @30 epoch						
Input gambar 100x100x3						
Configuration	Layers	Regularization	Train Error	Test Error	Train Accuracy	Val Accuracy
Model 1	conv2d 32x98x98, k=3, s=1, relu flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy		0.0023	1.2059	1.00	0.77
Model 2	conv2d 32x98x98, k=3, s=1, relu max_pooling2d k=3 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy		0.0156	0.8355	0.98	0.77
Model 3	conv2d 32x98x98, k=3, s=1, relu max_pooling2d k=3 conv2d 32x30x30, k=3, s=1, relu max_pooling2d k=3 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy		0.2024	0.4221	0.9196	0.8305
Model 4	conv2d 32x98x98, k=3, s=1, relu max_pooling2d k=3 conv2d 32x30x30, k=3, s=1, relu max_pooling2d k=3 conv2d 32x8x8, k=3, s=1, relu max_pooling2d k=3 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy		0.3090	0.4690	0.8656	0.7949
Model 5	conv2d 32x98x98, k=3, s=1, relu max_pooling2d k=3 conv2d 32x30x30, k=3, s=1, relu max_pooling2d k=3 conv2d 32x8x8, k=3, s=1, relu max_pooling2d k=3 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy	dropout = 0.2 dropout = 0.2 dropout = 0.2	0.4329	0.4418	0.8030	0.7884
Model 6	conv2d 32x98x98, k=3, s=1, relu max_pooling2d k=3 conv2d 64x30x30, k=3, s=1, relu max_pooling2d k=3 conv2d 128x8x8, k=3, s=1, relu max_pooling2d k=3 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy	dropout = 0.2 dropout = 0.2 dropout = 0.2	0.3933	0.3932	0.8213	0.8204
Model 7	conv2d 32x98x98, k=3, s=1, relu batch_normalization max_pooling2d k=3 conv2d 64x30x30, k=3, s=1, relu batch_normalization max_pooling2d k=3 conv2d 128x8x8, k=3, s=1, relu batch_normalization max_pooling2d k=3 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy	dropout = 0.2 dropout = 0.2 dropout = 0.2	0.2864	0.3492	0.8746	0.8430
Model 8	conv2d 32x98x98, k=3, s=1, relu batch_normalization max_pooling2d k=3 conv2d 64x30x30, k=3, s=1, relu	dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.5	0.3287	0.3096	0.8409	0.8682

	batch_normalization max_pooling2d k=3 conv2d 128x8x8, k=3, s=1, relu batch_normalization max_pooling2d k=3 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy callbacks = early stoping 'val_loss', p=5, restore best weights					
Model 9	conv2d 32x98x98, k=3, s=1, he_uniform,relu batch_normalization max_pooling2d k=3 conv2d 64x30x30, k=3, s=1, he_uniform, relu batch_normalization max_pooling2d k=3 conv2d 128x8x8, k=3, s=1, he_uniform, relu batch_normalization max_pooling2d k=3 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy callbacks = early stoping 'val_loss', p=5, restore best weights	dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.5	0.4366	0.3907	0.7983	0.8287
Model 10	conv2d 32x98x98, k=3, s=1, he_uniform,relu batch_normalization max_pooling2d k=3, s=2 conv2d 64x30x30, k=3, s=1, he_uniform, relu batch_normalization max_pooling2d k=3, s=2 conv2d 128x8x8, k=3, s=1, he_uniform, relu batch_normalization max_pooling2d k=3, s=2 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy callbacks = early stoping 'val_loss', p=5, restore best weights	dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.5	0.4250	0.4273	0.8043	0.8061
Model 11	conv2d 32x98x98, k=3, s=1, relu batch_normalization max_pooling2d k=2 conv2d 64x30x30, k=3, s=1, relu batch_normalization max_pooling2d k=2 conv2d 128x8x8, k=3, s=1, relu batch_normalization max_pooling2d k=2 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy callbacks = early stoping 'val_loss', p=5, restore best weights	dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.5	0.3843	0.4108	0.8260	0.8206
Model 12	conv2d 32x98x98, k=3, s=1, relu batch_normalization max_pooling2d k=2, s=2 conv2d 64x30x30, k=3, s=1, relu batch_normalization max_pooling2d k=2, s=2 conv2d 128x8x8, k=3, s=1, relu batch_normalization max_pooling2d k=2, s=2 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy callbacks = early stoping 'val_loss', p=5, restore best weights	dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.5	0.3628	0.4019	0.8399	0.8275
Model 13	conv2d 32x98x98, k=3, s=1, relu batch_normalization max_pooling2d k=2, s=2 conv2d 64x47x47, k=3, s=1, relu batch_normalization max_pooling2d k=2, s=2 conv2d 128x21x21, k=3, s=1, relu	dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.5	0.3994	0.4098	0.8182	0.8169

	batch_normalization max_pooling2d k=2, s=2 conv2d 256x8x8, k=3, s=1, relu batch_normalization max_pooling2d k=2, s=2 flatten dense 256 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy callbacks = early stoping 'val_loss', p=5, restore best weights					
Model 14	conv2d 32x98x98, k=3, s=1, relu batch_normalization max_pooling2d k=2 conv2d 64x30x30, k=3, s=1, relu batch_normalization max_pooling2d k=2 conv2d 128x8x8, k=3, s=1, relu batch_normalization max_pooling2d k=2 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy callbacks = early stoping 'val_loss', p=5, restore best weights	dropout = 0.2 dropout = 0.3 dropout = 0.4 dropout = 0.5	0.3633	0.3917	0.8342	0.8361
Model 15	conv2d 32x98x98, k=3, s=1, relu batch_normalization max_pooling2d k=2 conv2d 64x47x47, k=3, s=1, relu batch_normalization max_pooling2d k=2 conv2d 128x21x21, k=3, s=1, relu batch_normalization max_pooling2d k=2 conv2d 256x8x8, k=3, s=1, relu batch_normalization max_pooling2d k=2 flatten dense 512 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy callbacks = early stoping 'val_loss', p=5, restore best weights	dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.5	0.3019	0.4040	0.8678	0.8352
Model 16	conv2d 32x98x98, k=3, s=1, relu batch_normalization max_pooling2d k=3 conv2d 64x30x30, k=3, s=1, relu batch_normalization max_pooling2d k=3 conv2d 128x8x8, k=3, s=1, relu batch_normalization max_pooling2d k=3 flatten dense 128 ,relu dense 2, softmax sparse_categorical_crossentropy, SGD(lr = 0.001, momentum=0.9) accuracy callbacks = early stoping 'val_loss', p=5, restore best weights	dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.5 l2 = 0.001 l2 = 0.001 l2 = 0.001	0.4813	0.5165	0.8253	0.8143
Model 17	AUGMENTASI DATA (Image Data Generator) conv2d 32x98x98, k=3, s=1, relu batch_normalization max_pooling2d k=2 conv2d 64x47x47, k=3, s=1, relu batch_normalization max_pooling2d k=2 conv2d 128x21x21, k=3, s=1, relu batch_normalization max_pooling2d k=2 conv2d 256x8x8, k=3, s=1, relu batch_normalization max_pooling2d k=2 flatten dense 512 ,relu dense 1, sigmoid binary_crossentropy, adam, batch=128 accuracy	dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.2 dropout = 0.2	0.1400	0.1109	0.9431	0.9580

	callbacks = [early stoping 'val_loss', p=5, restore best weights],[ReduceLROnPlateau 'val_loss', p=3, factor=0.5, min_lr = 0.00001]					

Refrensi

Datasets
https://www.kaggle.com/competitions/dogs-vs-cats/data

code
https://www.kaggle.com/code/sachinpatil1280/cats-vs-dogs-image-classification-using-cnn-95/notebook
https://www.youtube.com/watch?v=FLf5qmSOkwU&t=1089s
https://www.youtube.com/watch?v=4ae13fiKDqo
https://www.youtube.com/watch?v=KmrR-ceL7d8
https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/