Tugas 1

IF4074 Pembelajaran Mesin Lanjutan

Milestone B : Back Propagation pada *Convolutional Neural Network*

Disusun oleh:

Kelompok 10

M. Rafli Zamzami	13519067
Aria Bachrul Ulum Berlian	13519115
Safiq Faray	13519145

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

A. Struktur Program

Program yang kami buat adalah implementasi *Convolutional Neural Network* (CNN) yang dapat melakukan training dengan mengimplementasikan proses *backpropagation*. Proses *backpropagation* yang diimplementasi menggunakan *library* numpy sebagai pendukung kalkulasi matriks. Selain itu, seluruh proses *backpropagation* diimplementasi secara manual mengikuti acuan dari slide kuliah serta artikel artikel lain seperti kalkulasi turunan dan sebagainya.

Proses pembuatan model dan operasi lainnya dilakukan pada file "main.py". Untuk implementasi CNN, kami letakkan pada folder "neuralnetwork" sebagai modul yang dapat dilakukan *import*.

Untuk backpropagation, kami juga mengimplementasikan *gradient clipping* untuk menghindari ledakan gradient dan/atau overflow saat melakukan operasi perhitungan gradient. Loss function yang kami gunakan adalah **binary cross entropy loss.**

Berikut adalah *class* yang kami buat untuk menyusun CNN.

a. Convolution Layer

Merupakan class "ConvLayer" yang telah dirancang pada milestone A. Terdapat penambahan fungsi backpropagation serta fungsi pendukung untuk mempermudah fitur *save/load*.

b. Pooling Layer

Merupakan class "Pooling" yang telah dirancang pada milestone A. Terdapat penambahan fungsi backpropagation serta fungsi pendukung untuk mempermudah fitur *save/load*.

c. Flatten Layer

Merupakan class "Flatten" pada milestone A dengan penambahan fungsi backpropagation.

d. Dense Layer

Merupakan class "Dense" yang telah dirancang pada milestone A. Terdapat penambahan fungsi backpropagation serta fungsi pendukung untuk mempermudah fitur *save/load*.

e. Activation

Merupakan class “Activation” pada milestone A dengan penambahan fungsi kalkulasi turunan.

f. Model

Merupakan class “NN” yang telah dirancang pada milestone A. Terdapat penambahan fungsi backpropagation serta fungsi pendukung untuk mempermudah fitur *save/load*. Model yang telah dibuat dapat disimpan dengan memanggil fungsi **save_model** dengan parameter nama file serta indentasi.

g. Trainer

Merupakan class “Trainer” yang berfungsi sebagai pengatur proses *training*. Inisialisasi class Trainer memerlukan parameter **model** sebagai model yang akan dilakukan training, **epoch** sebagai banyaknya iterasi dari seluruh testing data, **learning_rate**, **input**, serta **label**. Mengacu pada SOLID principle, seluruh proses backpropagation mulai dari kalkulasi turunan hingga update filter dilakukan pada kelas kelas lain. Class Trainer memiliki fungsi evaluasi untuk memperlihatkan hasil training.

h. Evaluator

Merupakan class “Evaluator” yang berfungsi sebagai evaluasi terhadap hasil prediksi dengan target label. Class Evaluator memiliki beberapa fungsi untuk menghitung matriks *accuracy*, *recall, precision,* serta *F1*. Parameter inisialisasi adalah **num_classes** yaitu banyaknya kelas, **labels** yaitu hasil prediksi dari forward propagation, serta **target_labels** yaitu target groundtruth.

i. Preprocess

Merupakan class “Preprocess” yang berfungsi untuk menyiapkan data pada folder “data”. Folder “data sendiri merupakan folder berisikan dataset pandas or bear.

B. Hasil

Pembelajaran dengan skema 90% data train dan 10% data test dilakukan dengan 100 data saja. Hal ini dikarenakan limitasi waktu dan perangkat keras. Arsitektur model yang digunakan adalah sebagai berikut.

```
image = cv2.imread("251.jpeg")
model = NN(image.shape)
model.add(
      ConvLayer(
      input_shape=image.shape,
      padding=0,
      num_filters=1,
      filter_size=(3, 3),
      stride=1,
      detector_function="relu",
      )
)
model.add(Pooling(mode="max", pool_size=(2, 2), stride=2))
model.add(Flatten())

flat_shape = model.layers[2].feature_map_shape
model.add(Dense(8, flat_shape, "relu"))

model.add(Dense(1, 8, "sigmoid"))
model.save_model("base.json", 4)
```

Lalu, berikut adalah kode untuk training dan hasil training. **Parameter training** adalah dengan **epoch 1** dan **learn rate 0.1**

```
data_preprocess = Preprocess("../data", False)
train, test = data_preprocess.get_data_split(split=(0.9,0.1),
max_data_len=100)

trainer = Trainer(
      model,
      1,
      0.1,
      train.get_images(),
      train.get_labels(),
      test_input=test.get_images(),
      test_label=test.get_labels(),
)
trainer.fit(save=True)
```

```
Training model...
Epoch 1/1
c:\DATA\Smt9\mll\tubes-cnn\src\neuralnet\Activation.py:49: RuntimeWarning: overflow encountered in exp
  return 1 / (1 + np.exp(-input))
Confusion Matrix:
[[ 0 42]
 [ 0 48]]
Accuracy: 53.33%
Average Precision: 26.67%
Average Recall: 50.00%
Average F1 Score: 34.78%
Training finished. Saving model...
```

Setelah proses training, model di **save** dalam bentuk **JSON**. berikut adalah potongan layer Convolution dan Dense output pada model sebelum dan sesudah training.

Base.json


```
yers": [
{"type": "conv2d", "input_shape": [256, 256, 3], "padding": 0, "num_filters": 1,
"filter_size": [3, 3], "stride": 1, "detector_function": {"name": "relu"}, "filter":
[[[-0.2564921094562205, 0.09560742360037879, 0.1417435783296569], [0.6411939423186458, -1.
3340125661371052, 1.1047233874350575], [0.01825428384329426, 0.2299682745168528, -0.
15884150816192133]]]},
 "biases": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]},
{"type": "dense", "num_units": 1, "detector_function": {"name": "sigmoid"}, "weights":
[[1.0438584537994662, 1.9799836947134757, -2.6064845376191474, 0.18602451688425886, -0.
019826793625509695, -1.3780030145353255, 1.2277760070787085, 0.08184800739320314]],
"biases": [0.0]}
```


Trained.json


```
ers": [
{"type": "conv2d", "input_shape": [256, 256, 3], "padding": 0, "num_filters": 1,
"filter_size": [3, 3], "stride": 1, "detector_function": {"name": "relu"}, "filter": [[[3.
2435078905437815, 3.3956074236003806, 3.8417435783296594], [3.941193942318648, 2.
1659874338628957, 4.804723387435057], [3.518254283843296, 3.729968274516855, 3.
5411584918380807]]]},
{"type": "dense", "num_units": 1, "detector_function": {"name": "sigmoid"}, "weights":
[[0.04385845379946632, 0.07998369471347505, -1.7064845376191466, 0.08602451688425886, 0.
08017320637449031, 0.02199698546467463, 1.1277760070787084, 0.08184800739320314]],
"biases": [0.1]}
```


Setelah model di save dalam bentuk JSON, model tersebut bisa di **load** dengan kode di bawah ini.

```
newModel = load_model("base.json")
```

Berikut adalah contoh inferensi menggunakan model hasil latih.

```
# take 10 data from test set and predict
test_images = test.get_images()[:10]
test_labels = test.get_labels()[:10]
pred = model.predict(test_images)
id2label = {0:"bear" , 1 :"panda"}

for i in range(len(pred)):
```

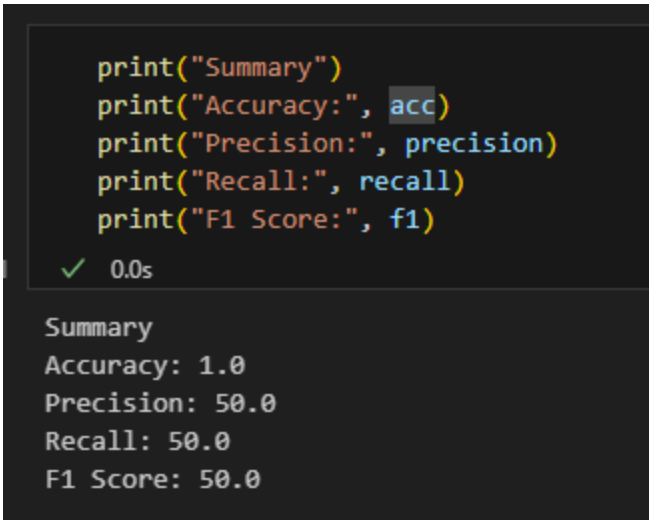
```
      print("Predicted: ", id2label[pred[i]], "Actual: ",
id2label[test_labels[i][0]])
```

```
Predicted:  panda Actual:  panda
Predicted:  panda Actual:  panda
Predicted:  panda Actual:  panda
Predicted:  panda Actual:  panda
Predicted:  panda Actual:  bear
Predicted:  panda Actual:  bear
Predicted:  panda Actual:  panda
Predicted:  panda Actual:  bear
Predicted:  panda Actual:  bear
Predicted:  panda Actual:  bear
```

Untuk eksperimen **10-fold cross validation** berikut kode setup dan hasilnya.

```
def combine_datasets(datasets:list):
      combined = ImageDataset([])
      for dataset in datasets:
      combined.data += dataset.data
      return combined

evaluations = []
for i in range(len(data_kfold)):
      print("Fold", i+1)
      exclusion = [data for j, data in enumerate(data_kfold) if j
!= i]
      train_data = combine_datasets(exclusion)
      test_data = data_kfold[i]
      trainer = Trainer(
      newModel,
      1,
      0.1,
      train_data.get_images(),
      train_data.get_labels(),
      test_input=test_data.get_images(),
      test_label=test_data.get_labels(),
      )
      eval = trainer.fit(save=False)
      print()
      evaluations.append(eval)
```



Terdapat kegagalan eksperimen yang mungkin pada setup/pembagian data nya. Hal ini disimpulkan dari hasil percobaan yang menghasilkan 100% accuracy, yang seharusnya tidak mungkin pada data yang kecil.

C. Pembagian Tugas

Anggota Kelompok	**Pembagian Tugas**
M, Rafli Zamzami (13519067)	Backpropagation pada dense layer, class trainer, save load
Aria Bachrul Ulum Berlian (13519115)	Backpropagation pada pooling layer, experiment, class evaluator
Safiq Faray (13519145)	Backpropagation pada convolution layer, derivative activation function, 10-fold cross validation