TUGAS LAB SESSION 3
PEMELAJARAN MESIN LANJUT GANJIL 2020/2021

Nama : Marshal Arijona Sinaga
NPM : 2006560983

Keterangan:

- semua komputasi pada cell dilakukan pada environment cloud Floydhub
- Training dilakukan dengan menggunakan GPU Tesla V80
- Untuk menjalankan komputasi, lakukan penyesuaian terhadap folder untuk menyimpan dataset
- Pada folder zip terdapat file parameter model awal, 3 file parameter model skenario dan 3 file parameter model pretrained yang dapat digunakan.

TRANSFER LEARNING

1. Bangun sebuah model CNN dengan arsitektur dasar sebagai berikut :
    - Convolutional Layer (25 maps, kernel 3x3)
    - Pooling Layer (2x2) - Activation Function (ReLU function)
    - Convolutional Layer (50 maps, kernel 3x3)
    - Pooling Layer (2x2)
    - Activation Function (ReLU function)
    - Convolutional Layer (100 maps, kernel 3x3)
    - Pooling Layer (2x2)
    - Activation Function (ReLU function)
    - Hidden Layer (100 neuron)
    - Activation Function (ReLU function)
    - Output Layer (10 kelas)
    - Activation Function (Softmax function)
    - Classification Result

    Dengan tambahan beberapa seperti :
    - Adam optimization
    - Early Stop

```python
class network_1(nn.Module):
    def __init__(self):
        super(network_1, self).__init__()
        self.convolutional1 = nn.Conv2d(3, 25, kernel_size=3, stride=1, padding=1)
        self.maxpool1 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.relu1 = nn.ReLU()
        self.convolutional2 = nn.Conv2d(25, 50, kernel_size=3, stride=1, padding=1)
        self.maxpool2 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.relu2 = nn.ReLU()
        self.convolutional3 = nn.Conv2d(50, 100, kernel_size=3, stride=1, padding=1)
        self.maxpool3 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.relu3 = nn.ReLU()
        self.linear1 = nn.Linear(FLATTEN_SIZE_1, 100)
        self.relu4 = nn.ReLU()
        self.linear2 = nn.Linear(100, 10)
        self.softmax1 = nn.Softmax(dim=1)



    def forward(self, x):
        x = self.convolutional1(x)
        x = self.maxpool1(x)
        x = self.relu1(x)
        x = self.convolutional2(x)
        x = self.maxpool2(x)
        x = self.relu2(x)
        x = self.convolutional3(x)
        x = self.maxpool3(x)
        x = self.relu3(x)
        x = x.view(-1, FLATTEN_SIZE_1)
        x = self.linear1(x)
        x = self.relu4(x)
        x = self.linear2(x)
        x = self.softmax1(x)

        return x
```

Untuk classfication result diimplementasikan secara implisit pada saat proses training, testing, maupun validaition. Classification result sendiri berupa fungsi yang mencari nilai maksimal dari komponen softmax output.

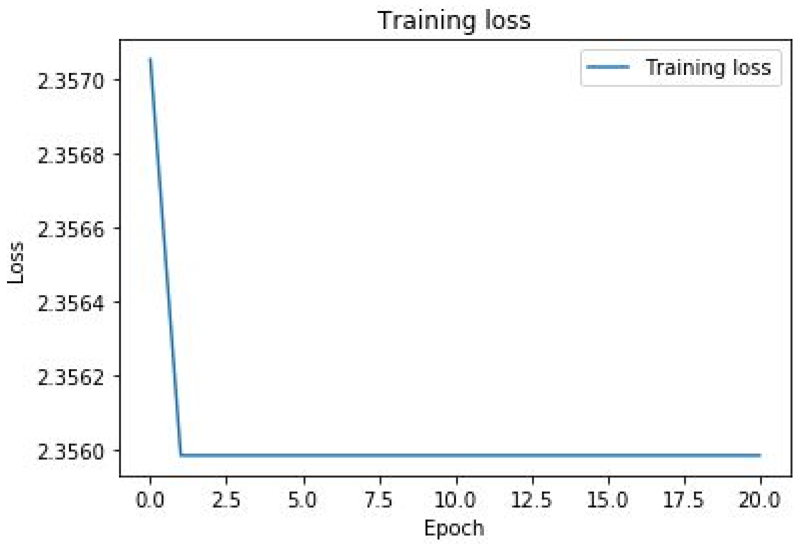
2. Implementasi model CNN tersebut pada dataset CIFAR 10 (ambil random hanya 1000 citra per kelas) dengan inisial epoch = 100, batch number = 10, learning rate= 0.1, serta ratio data train dan data test sebanyak 60% : 40%.

Jawab:

- Pertama, download dataset yang akan digunakan yaitu CIFAR10. Selanjutnya, pilih secara acak indeks dari dataset yang nantinya akan digunakan untuk proses training dan testing
- Untuk fase training, karena menggunakan early stop, data training dibagi lagi menjadi validation set dan testing set. Dari 4000 data training, dibagi menjadi 3000 data untuk training dan 1000 data untuk validation
- Setelah itu, inisialisasi model yang akan dilatih. Inisialisasi terdiri dari penentuan optimator (sesuai ketentuan soal), loss function (meminimalkan cross entropy function), dan menginisialisasi early stop. Early stop yang digunakan mengadaptasi

model early stop yang dikembangkan oleh Stefano Nardo (https://gist.github.com/stefanonardo/693d96ceb2f531fa05db530f3e21517d). Early stop terdiri dari beberapa parameter yaitu: patience yang merupakan jumlah epoch yang dapat ditoleransi bila loss function tidak mengalami perbaikan, mode dari early stop (min atau max) yang menjadi kriteria yang menentukan loss function mana yang lebih baik (apakah lebih besar atau lebih kecil), serta nilai epsilon yang menjadi nilai toleransi perhitungan loss function yang lebih baik. Pada tugas ini, metrik yang digunakan untuk menentukan early stop adalah loss function (cross entropy loss function). Early stop criterion dinilai berdasarkan nilai cross entropy loss pada validation set. Apabila kriteria tersebut terpenuhi, maka proses training akan berhenti.

- Berikut ini adalah hasil pelatihan model.
  Epoch-0: Average-cross entropy Loss:2.357053814729055
  Epoch-2: Average-cross entropy Loss:2.355983680089315
  Epoch-4: Average-cross entropy Loss:2.355983672142029
  Epoch-6: Average-cross entropy Loss:2.355983681678772
  Epoch-8: Average-cross entropy Loss:2.3559836741288502
  Epoch-10: Average-cross entropy Loss:2.355983680486679
  Epoch-12: Average-cross entropy Loss:2.355983679294586
  Epoch-14: Average-cross entropy Loss:2.3559836769104003
  Epoch-16: Average-cross entropy Loss:2.3559836769104003
  Epoch-18: Average-cross entropy Loss:2.355983677705129
  Epoch-20: Average-cross entropy Loss:2.355983678499858
  Early stopping on epoch-20
  Epoch-20: Average-cross entropy Loss:2.355983678499858
- Log tersebut menunjukkan nilai average loss function pada training set. Terlihat tidak ada peningkatan yang signifikan terhadap loss dari model. Pada epoch ke-20 model mengalami early stopping. Pada tugas ini, dipilih nilai toleransi epoch sebesar 15, dengan mode min dan nilai toleransi 1e-4.
- Berikut ini adalah grafik dari loss function tersebut:



3. Lakukan skenario eksperimen untuk mendapatkan metrik evaluasi optimal (akurasi, sensitivity, specifity, dan f1-score) dengan mengubah parameter yaitu:
(a) Mengubah epoch, dengan learning rate = 0.1 dan batch number = 10
Jawab:

Epoch diubah dari 100 menjadi 120.

Epoch-0: Average-cross entropy Loss:2.360725847085317
Epoch-2: Average-cross entropy Loss:2.360150343179703
Epoch-4: Average-cross entropy Loss:2.3601503483454387
Epoch-6: Average-cross entropy Loss:2.3601503439744316
Epoch-8: Average-cross entropy Loss:2.360150342384974
Epoch-10: Average-cross entropy Loss:2.3601503415902454
Epoch-12: Average-cross entropy Loss:2.3601503443717955
Epoch-14: Average-cross entropy Loss:2.360150343577067
Epoch-16: Average-cross entropy Loss:2.3601503479480743
Epoch-18: Average-cross entropy Loss:2.3601503419876098
Epoch-20: Average-cross entropy Loss:2.3601503427823385
Epoch-22: Average-cross entropy Loss:2.3601503400007884
Epoch-24: Average-cross entropy Loss:2.3601503481467563
Epoch-26: Average-cross entropy Loss:2.3601503495375313
Epoch-28: Average-cross entropy Loss:2.3601503415902454
Epoch-30: Average-cross entropy Loss:2.36015034755071
Epoch-32: Average-cross entropy Loss:2.360150343179703
Epoch-34: Average-cross entropy Loss:2.3601503449678423
Epoch-36: Average-cross entropy Loss:2.36015034755071
Epoch-38: Average-cross entropy Loss:2.360150348742803
Epoch-40: Average-cross entropy Loss:2.3601503483454387
Epoch-42: Average-cross entropy Loss:2.360150343179703
Epoch-44: Average-cross entropy Loss:2.36015034476916
Epoch-46: Average-cross entropy Loss:2.3601503403981527
Epoch-48: Average-cross entropy Loss:2.3601503443717955
Epoch-50: Average-cross entropy Loss:2.3601503419876098
Epoch-52: Average-cross entropy Loss:2.360150343179703
Epoch-54: Average-cross entropy Loss:2.360150348742803
Epoch-56: Average-cross entropy Loss:2.3601503485441206
Epoch-58: Average-cross entropy Loss:2.3601503519217175
Epoch-60: Average-cross entropy Loss:2.3601503439744316
Epoch-62: Average-cross entropy Loss:2.3601503427823385
Epoch-64: Average-cross entropy Loss:2.360150345961253
Epoch-66: Average-cross entropy Loss:2.3601503403981527
Epoch-68: Average-cross entropy Loss:2.3601503471533456
Epoch-70: Average-cross entropy Loss:2.3601503455638886
Epoch-72: Average-cross entropy Loss:2.3601503499348957
Epoch-74: Average-cross entropy Loss:2.36015034476916
Epoch-76: Average-cross entropy Loss:2.3601503495375313
Epoch-78: Average-cross entropy Loss:2.3601503403981527
Epoch-80: Average-cross entropy Loss:2.3601503415902454
Epoch-82: Average-cross entropy Loss:2.3601503455638886
Epoch-84: Average-cross entropy Loss:2.3601503439744316
Epoch-86: Average-cross entropy Loss:2.3601503401994703
Epoch-88: Average-cross entropy Loss:2.3601503455638886
Epoch-90: Average-cross entropy Loss:2.3601503467559812
Epoch-92: Average-cross entropy Loss:2.3601503519217175
Epoch-94: Average-cross entropy Loss:2.3601503471533456

Epoch-96: Average-cross entropy Loss:2.3601503485441206
Epoch-98: Average-cross entropy Loss:2.36015035033226
Epoch-100: Average-cross entropy Loss:2.3601503439744316
Epoch-102: Average-cross entropy Loss:2.360150345166524
Epoch-104: Average-cross entropy Loss:2.3601503400007884
Epoch-106: Average-cross entropy Loss:2.3601503469546636
Epoch-108: Average-cross entropy Loss:2.3601503471533456
Epoch-110: Average-cross entropy Loss:2.3601503415902454
Epoch-112: Average-cross entropy Loss:2.36015034476916
Epoch-114: Average-cross entropy Loss:2.360150345166524
Epoch-116: Average-cross entropy Loss:2.360150343577067
Epoch-118: Average-cross entropy Loss:2.3601503443717955

Nilai loss dari model tidak berubah mulai dari epoch 2. Nilai loss cenderung statis hingga epoch ke 120

(b) Mengubah jumlah batch number dengan epoch optimal dari skenario (a) dan learning rate = 0.1

Jawab:

Pada skenario ini, nilai epoch diubah menjadi 120. Meskipun pada percobaan sebelumnya, perubahan nilai epoch tidak memberi hasil yang lebih baik, namun penggunaan epoch yang lebih banyak umumnya memberi hasil yang lebih baik. Untuk ukuran minibatch yang dipilih adalah 100. Berikut ini adalah log dari proses training dari skenario 2:

Epoch-0: Average-cross entropy Loss:2.3620065371195476
Epoch-2: Average-cross entropy Loss:2.362484077612559
Epoch-4: Average-cross entropy Loss:2.362484089533488
Epoch-6: Average-cross entropy Loss:2.362484089533488
Epoch-8: Average-cross entropy Loss:2.362484085559845
Epoch-10: Average-cross entropy Loss:2.362484085559845
Epoch-12: Average-cross entropy Loss:2.3624840935071307
Epoch-14: Average-cross entropy Loss:2.362484085559845
Epoch-16: Average-cross entropy Loss:2.362484097480774
Epoch-18: Average-cross entropy Loss:2.362484097480774
Epoch-20: Average-cross entropy Loss:2.362484085559845
Epoch-22: Average-cross entropy Loss:2.362484085559845
Epoch-24: Average-cross entropy Loss:2.362484089533488
Epoch-26: Average-cross entropy Loss:2.362484077612559
Epoch-28: Average-cross entropy Loss:2.362484097480774
Epoch-30: Average-cross entropy Loss:2.362484081586202
Epoch-32: Average-cross entropy Loss:2.362484077612559
Epoch-34: Average-cross entropy Loss:2.362484081586202
Epoch-36: Average-cross entropy Loss:2.362484081586202
Epoch-38: Average-cross entropy Loss:2.362484085559845
Epoch-40: Average-cross entropy Loss:2.3624840935071307
Epoch-42: Average-cross entropy Loss:2.362484101454417
Epoch-44: Average-cross entropy Loss:2.362484089533488
Epoch-46: Average-cross entropy Loss:2.362484077612559

Epoch-48: Average-cross entropy Loss:2.362484081586202
Epoch-50: Average-cross entropy Loss:2.362484085559845
Epoch-52: Average-cross entropy Loss:2.362484089533488
Epoch-54: Average-cross entropy Loss:2.362484085559845
Epoch-56: Average-cross entropy Loss:2.362484101454417
Epoch-58: Average-cross entropy Loss:2.362484101454417
Epoch-60: Average-cross entropy Loss:2.362484085559845
Epoch-62: Average-cross entropy Loss:2.362484073638916
Epoch-64: Average-cross entropy Loss:2.362484085559845
Epoch-66: Average-cross entropy Loss:2.3624840935071307
Epoch-68: Average-cross entropy Loss:2.362484085559845
Epoch-70: Average-cross entropy Loss:2.362484077612559
Epoch-72: Average-cross entropy Loss:2.362484081586202
Epoch-74: Average-cross entropy Loss:2.362484085559845
Epoch-76: Average-cross entropy Loss:2.362484085559845
Epoch-78: Average-cross entropy Loss:2.362484081586202
Epoch-80: Average-cross entropy Loss:2.362484085559845
Epoch-82: Average-cross entropy Loss:2.362484089533488
Epoch-84: Average-cross entropy Loss:2.362484085559845
Epoch-86: Average-cross entropy Loss:2.362484089533488
Epoch-88: Average-cross entropy Loss:2.362484081586202
Epoch-90: Average-cross entropy Loss:2.362484097480774
Epoch-92: Average-cross entropy Loss:2.362484085559845
Epoch-94: Average-cross entropy Loss:2.362484085559845
Epoch-96: Average-cross entropy Loss:2.362484085559845
Epoch-98: Average-cross entropy Loss:2.362484081586202
Epoch-100: Average-cross entropy Loss:2.362484085559845
Epoch-102: Average-cross entropy Loss:2.362484085559845
Epoch-104: Average-cross entropy Loss:2.362484077612559
Epoch-106: Average-cross entropy Loss:2.362484085559845
Epoch-108: Average-cross entropy Loss:2.362484081586202
Epoch-110: Average-cross entropy Loss:2.362484089533488
Epoch-112: Average-cross entropy Loss:2.362484101454417
Epoch-114: Average-cross entropy Loss:2.362484101454417
Epoch-116: Average-cross entropy Loss:2.362484081586202
Epoch-118: Average-cross entropy Loss:2.362484089533488

(c) Mengubah learning rate dengan epoch optimal dari skenario (a) dan jumlah batch number optimal dari skenario (b).

Pada skenario ini epoch yang dipilih adalah 120, ukuran minibatch bernilai 100, dan learning rate bernilai 0.0001. Berikut ini adalah log dari training phase:
Epoch-0: Average-cross entropy Loss:2.298665543397268
Epoch-2: Average-cross entropy Loss:2.183494202295939
Epoch-4: Average-cross entropy Loss:2.1558603286743163
Epoch-6: Average-cross entropy Loss:2.1317097226778667
Epoch-8: Average-cross entropy Loss:2.1200111071268717
Epoch-10: Average-cross entropy Loss:2.1086010098457337
Epoch-12: Average-cross entropy Loss:2.094999059041341

Epoch-14: Average-cross entropy Loss:2.088266011079152
Epoch-16: Average-cross entropy Loss:2.0796248535315196
Epoch-18: Average-cross entropy Loss:2.074263139565786
Epoch-20: Average-cross entropy Loss:2.0674545049667357
Epoch-22: Average-cross entropy Loss:2.058255926767985
Epoch-24: Average-cross entropy Loss:2.05321271220843
Epoch-26: Average-cross entropy Loss:2.0467239797115324
Epoch-28: Average-cross entropy Loss:2.039867111047109
Epoch-30: Average-cross entropy Loss:2.035265237092972
Epoch-32: Average-cross entropy Loss:2.028705859184265
Epoch-34: Average-cross entropy Loss:2.0227427383263907
Epoch-36: Average-cross entropy Loss:2.0106814404328666
Epoch-38: Average-cross entropy Loss:2.0075834810733797
Epoch-40: Average-cross entropy Loss:2.0073580483595532
Epoch-42: Average-cross entropy Loss:1.9965627511342368
Epoch-44: Average-cross entropy Loss:1.9921822011470796
Epoch-46: Average-cross entropy Loss:1.9834979097048442
Epoch-48: Average-cross entropy Loss:1.982373547554016
Epoch-50: Average-cross entropy Loss:1.9720630764961242
Epoch-52: Average-cross entropy Loss:1.9680478473504384
Epoch-54: Average-cross entropy Loss:1.9679739892482757
Epoch-56: Average-cross entropy Loss:1.9584186216195425
Epoch-58: Average-cross entropy Loss:1.9550737142562866
Epoch-60: Average-cross entropy Loss:1.9492841919263204
Epoch-62: Average-cross entropy Loss:1.9469713826974233
Epoch-64: Average-cross entropy Loss:1.9405913054943085
Epoch-66: Average-cross entropy Loss:1.9352835834026336
Epoch-68: Average-cross entropy Loss:1.9351104776064554
Epoch-70: Average-cross entropy Loss:1.9292508661746979
Epoch-72: Average-cross entropy Loss:1.9246251066525777
Epoch-74: Average-cross entropy Loss:1.9215600172678629
Epoch-76: Average-cross entropy Loss:1.9191421250502267
Epoch-78: Average-cross entropy Loss:1.9162919123967488
Epoch-80: Average-cross entropy Loss:1.911865496635437
Epoch-82: Average-cross entropy Loss:1.9091520388921102
Epoch-84: Average-cross entropy Loss:1.9052610794703166
Epoch-86: Average-cross entropy Loss:1.9009269972642262
Epoch-88: Average-cross entropy Loss:1.8996048291524252
Epoch-90: Average-cross entropy Loss:1.8969434559345246
Epoch-92: Average-cross entropy Loss:1.8975771069526672
Epoch-94: Average-cross entropy Loss:1.8896552900473276
Epoch-96: Average-cross entropy Loss:1.8910531838734945
Epoch-98: Average-cross entropy Loss:1.8837517062822977
Epoch-100: Average-cross entropy Loss:1.8827872395515441
Epoch-102: Average-cross entropy Loss:1.8827334662278494
Epoch-104: Average-cross entropy Loss:1.876766667763392
Epoch-106: Average-cross entropy Loss:1.8747148712476094
Epoch-108: Average-cross entropy Loss:1.8688309808572134
Epoch-110: Average-cross entropy Loss:1.8708036959171295
Epoch-112: Average-cross entropy Loss:1.8661884824434916

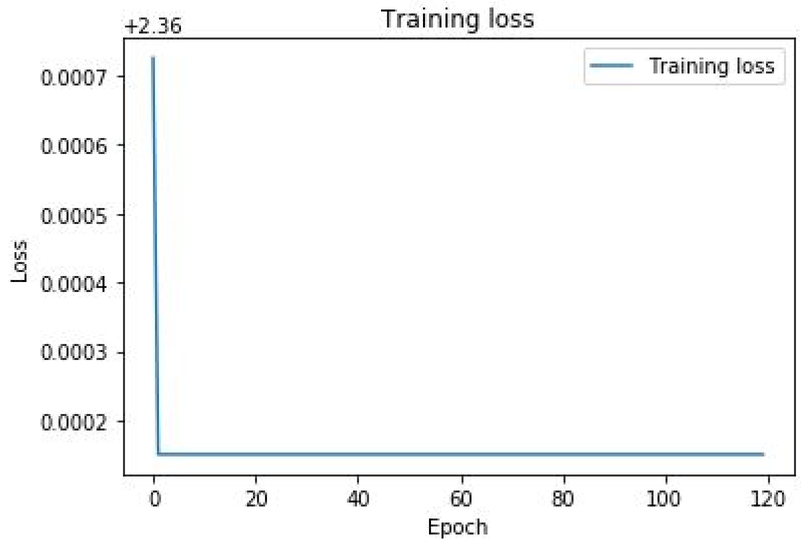
Epoch-114: Average-cross entropy Loss:1.8622583766778311
Epoch-116: Average-cross entropy Loss:1.859093685944875
Epoch-118: Average-cross entropy Loss:1.8580187499523162
Nilai learning rate yang lebih rendah menyebabkan step size gradient descent lebih kecil sehingga proses optimisasi lebih presisi. Hal ini menyebabkan model lebih mudah mencapai titik konvergen.

4. Simpan checkpoint dari tiap interval epoch dan model final dari proses training dari model CNN dengan parameter optimal yang diperoleh dari skenario agar bisa digunakan kembali untuk proses training selanjutnya.

```
'''
save the model the path
'''
def save_checkpoint(model, path):
    torch.save(model.state_dict(), path)
```

5. Visualisasikan error dari proses training yang diperoleh dari tiap epoch dan tampilkan matrik evaluasi (akurasi, sensitivity, specificity, dan f1-score) dari data test.

a. skenario a (epoch = 120, minibatch=10, learning rate = 0.1):



Accuracy: 0.09833

Average sensitivity: 0.009833333005555566

Sensitivity for each class:

class 0: 0.0

class 1: 0.09833

class 2: 0.0

class 3: 0.0

class 4: 0.0

class 5: 0.0

class 6: 0.0

class 7: 0.0

class 8: 0.0

class 9: 0.0

Average specificity: 0.8098333063388898

Specificity for each class:

class 0: 0.8903

class 1: 0.0

class 2: 0.9113

class 3: 0.892

class 4: 0.9167

class 5: 0.893

class 6: 0.901

class 7: 0.9033

class 8: 0.8963

class 9: 0.8943

Average f1-score: 0.017905918057663124

f1-score for each class:

class 0: 0.0

class 1: 0.1791

class 2: 0.0

class 3: 0.0

class 4: 0.0

class 5: 0.0

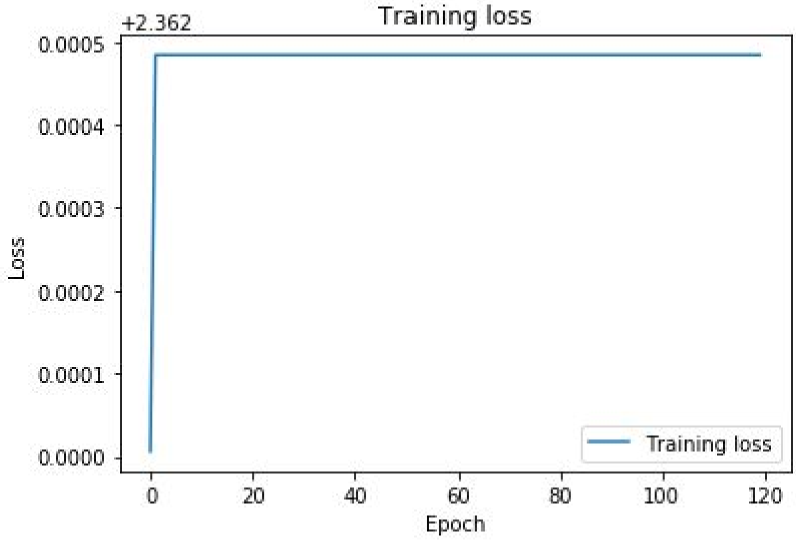
class 6: 0.0

class 7: 0.0

class 8: 0.0

class 9: 0.0

b. Skenario b (epoch = 120, minibatch = 100, learning rate = 0.1):



Accuracy: 0.1057

Average sensitivity: 0.010566666314444455

Sensitivity for each class:

class 0: 0.0

class 1: 0.0

class 2: 0.0

class 3: 0.0

class 4: 0.0

class 5: 0.0

class 6: 0.0

class 7: 0.0

class 8: 0.0

class 9: 0.1057

Average specificity: 0.8105666396477785

Specificity for each class:

class 0: 0.8903

class 1: 0.9017

class 2: 0.9113

class 3: 0.892

class 4: 0.9167

class 5: 0.893

class 6: 0.901

class 7: 0.9033

class 8: 0.8963

class 9: 0.0

Average f1-score: 0.019113656918902624

f1-score for each class:

class 0: 0.0

class 1: 0.0

class 2: 0.0

class 3: 0.0

class 4: 0.0

class 5: 0.0

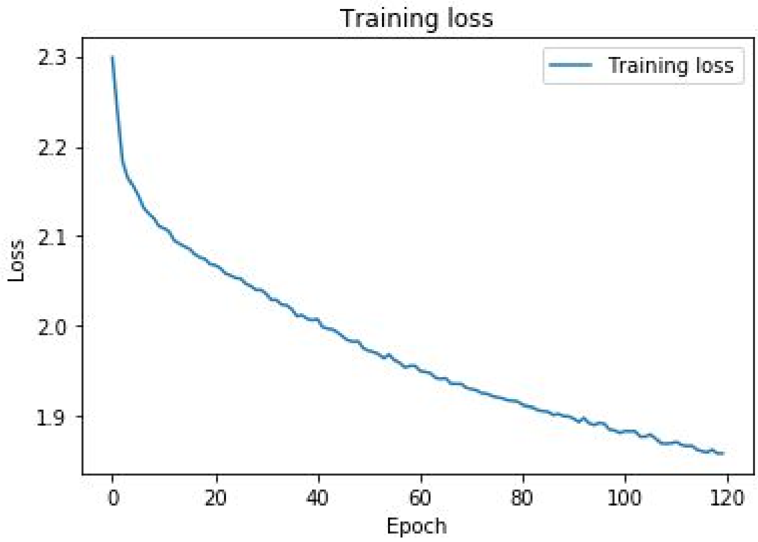
class 6: 0.0

class 7: 0.0

class 8: 0.0

class 9: 0.1911

c. skenario c (epoch = 120, minibatch = 100, learning rate = 0.0001):



Accuracy: 0.4767

Average sensitivity: 0.47734471156850977

Sensitivity for each class:

class 0: 0.6404

class 1: 0.5549

class 2: 0.3452

class 3: 0.3067

class 4: 0.3651

class 5: 0.4643

class 6: 0.5041

class 7: 0.4711

class 8: 0.5785

class 9: 0.5433

Average specificity: 0.9420219182207287

Specificity for each class:

class 0: 0.934

class 1: 0.956

class 2: 0.9266

class 3: 0.9215

class 4: 0.9497

class 5: 0.9218

class 6: 0.9578

class 7: 0.9495

class 8: 0.954

class 9: 0.9493

Average f1-score: 0.4699909496332613

f1-score for each class:

class 0: 0.5242

class 1: 0.5765

class 2: 0.2673

class 3: 0.3343

class 4: 0.4071

class 5: 0.3817

class 6: 0.5586

class 7: 0.5008

class 8: 0.5912

class 9: 0.5583

6. Gunakan pre-trained model yg telah dibuat tersebut untuk diimplementasikan pada dataset CIFAR 100 untuk kelas 1 hingga kelas 10 (masing-masing kelas terdiri dari 100 citra) dengan rasio data train dan data test sebanyak 60% : 40%. Kemudian hitung matrik evaluasi (akurasi, sensitivity, specificity, dan f1-score) dari data train dan data test.

Jawab:

- Pertama-tama dilakukan sampling ulang untuk mendapatkan dataset sesuai dengan deskripsi yang diminta. Setelah itu dilakukan training dan fine tuning terhadap model

dari skenario 3c karena memberi hasil yang paling baik dibandingkan dengan skenario lainnya. Berikut adalah log dari model pretraining tersebut:
Epoch-0: Average-cross entropy Loss:2.3216296831766763
Epoch-2: Average-cross entropy Loss:2.2730205059051514
Epoch-4: Average-cross entropy Loss:2.2526535193125405
Epoch-6: Average-cross entropy Loss:2.2377266883850098
Epoch-8: Average-cross entropy Loss:2.2269545793533325
Epoch-10: Average-cross entropy Loss:2.219796816507975
Epoch-12: Average-cross entropy Loss:2.2134652137756348
Epoch-14: Average-cross entropy Loss:2.207617441813151
Epoch-16: Average-cross entropy Loss:2.2022225856781006
Epoch-18: Average-cross entropy Loss:2.1967601776123047
Epoch-20: Average-cross entropy Loss:2.1891836325327554
Epoch-22: Average-cross entropy Loss:2.1877360741297402
Epoch-24: Average-cross entropy Loss:2.1834348837534585
Epoch-26: Average-cross entropy Loss:2.1766148805618286
Epoch-28: Average-cross entropy Loss:2.173719565073649
Epoch-30: Average-cross entropy Loss:2.170170863469442
Epoch-32: Average-cross entropy Loss:2.1683870951334634
Epoch-34: Average-cross entropy Loss:2.1660331090291343
Epoch-36: Average-cross entropy Loss:2.1620606184005737
Epoch-38: Average-cross entropy Loss:2.1616832415262857
Epoch-40: Average-cross entropy Loss:2.1584552923838296
Epoch-42: Average-cross entropy Loss:2.1559896071751914
Epoch-44: Average-cross entropy Loss:2.154154062271118
Epoch-46: Average-cross entropy Loss:2.1526780128479004
Epoch-48: Average-cross entropy Loss:2.1518653631210327
Epoch-50: Average-cross entropy Loss:2.150447368621826
Epoch-52: Average-cross entropy Loss:2.149410128593445
Epoch-54: Average-cross entropy Loss:2.148247321446737
Epoch-56: Average-cross entropy Loss:2.14697003364563
Epoch-58: Average-cross entropy Loss:2.144774834314982
Epoch-60: Average-cross entropy Loss:2.146044373512268
Epoch-62: Average-cross entropy Loss:2.1435721715291343
Epoch-64: Average-cross entropy Loss:2.1426618893941245
Epoch-66: Average-cross entropy Loss:2.141256888707479
Epoch-68: Average-cross entropy Loss:2.13860289255778
Epoch-70: Average-cross entropy Loss:2.136296113332113
Epoch-72: Average-cross entropy Loss:2.1345558563868203
Epoch-74: Average-cross entropy Loss:2.1323442459106445
Epoch-76: Average-cross entropy Loss:2.1280461152394614
Epoch-78: Average-cross entropy Loss:2.118884245554606
Epoch-80: Average-cross entropy Loss:2.116641124089559
Epoch-82: Average-cross entropy Loss:2.111540595690409
Epoch-84: Average-cross entropy Loss:2.106517573197683
Epoch-86: Average-cross entropy Loss:2.102960189183553
Epoch-88: Average-cross entropy Loss:2.097192406654358
Epoch-90: Average-cross entropy Loss:2.0931645035743713
Epoch-92: Average-cross entropy Loss:2.0905892054239907
Epoch-94: Average-cross entropy Loss:2.089136521021525

Epoch-96: Average-cross entropy Loss:2.0875069300333657
Epoch-98: Average-cross entropy Loss:2.0858190059661865
Epoch-100: Average-cross entropy Loss:2.084831714630127
Epoch-102: Average-cross entropy Loss:2.083991209665934
Epoch-104: Average-cross entropy Loss:2.08296807607015
Epoch-106: Average-cross entropy Loss:2.082165757815043
Epoch-108: Average-cross entropy Loss:2.0811362663904824
Epoch-110: Average-cross entropy Loss:2.0788148244222007
Epoch-112: Average-cross entropy Loss:2.076065460840861
Epoch-114: Average-cross entropy Loss:2.0747296810150146
Epoch-116: Average-cross entropy Loss:2.073683778444926
Epoch-118: Average-cross entropy Loss:2.0723000168800354
Epoch-120: Average-cross entropy Loss:2.0719969669977822
Epoch-122: Average-cross entropy Loss:2.0705196062723794
Epoch-124: Average-cross entropy Loss:2.0692118803660073
Epoch-126: Average-cross entropy Loss:2.0683038234710693
Epoch-128: Average-cross entropy Loss:2.068075875441233
Epoch-130: Average-cross entropy Loss:2.0670339465141296
Epoch-132: Average-cross entropy Loss:2.0670597155888877
Epoch-134: Average-cross entropy Loss:2.0661025842030845
Epoch-136: Average-cross entropy Loss:2.066350221633911
Epoch-138: Average-cross entropy Loss:2.0652334690093994
Epoch-140: Average-cross entropy Loss:2.0644739071528115
Epoch-142: Average-cross entropy Loss:2.0639802614847818
Epoch-144: Average-cross entropy Loss:2.0631409088770547
Epoch-146: Average-cross entropy Loss:2.06231951713562
Epoch-148: Average-cross entropy Loss:2.0617394049962363
Epoch-150: Average-cross entropy Loss:2.0607899030049643
Epoch-152: Average-cross entropy Loss:2.060288429260254
Epoch-154: Average-cross entropy Loss:2.0590640703837075
Epoch-156: Average-cross entropy Loss:2.058520476023356
Epoch-158: Average-cross entropy Loss:2.0584226846694946
Epoch-160: Average-cross entropy Loss:2.0580779711405435
Epoch-162: Average-cross entropy Loss:2.057701826095581
Epoch-164: Average-cross entropy Loss:2.0574092070261636
Epoch-166: Average-cross entropy Loss:2.057677070299784
Epoch-168: Average-cross entropy Loss:2.0565391778945923
Epoch-170: Average-cross entropy Loss:2.0548876921335855
Epoch-172: Average-cross entropy Loss:2.054404099782308
Epoch-174: Average-cross entropy Loss:2.0536529620488486
Epoch-176: Average-cross entropy Loss:2.05181348323822
Epoch-178: Average-cross entropy Loss:2.0504717032114663
Epoch-180: Average-cross entropy Loss:2.0492954651514688
Epoch-182: Average-cross entropy Loss:2.0482640663782754
Epoch-184: Average-cross entropy Loss:2.0479945739110312
Epoch-186: Average-cross entropy Loss:2.0481515924135842
Epoch-188: Average-cross entropy Loss:2.04768039782842
Epoch-190: Average-cross entropy Loss:2.046973963578542
Epoch-192: Average-cross entropy Loss:2.0464364687601724
Epoch-194: Average-cross entropy Loss:2.045854071776072

Epoch-196: Average-cross entropy Loss:2.0452797015508017
Epoch-198: Average-cross entropy Loss:2.0448131561279297
Berikut ini adalah grafik dari training loss:

Berikut ini adalah hasil evaluasi training set:
Accuracy: 0.2533

Average sensitivity: 0.1827633196440358

Sensitivity for each class:

class 0: 0.0

class 1: 0.3833

class 2: 0.0

class 3: 0.1797

class 4: 0.3

class 5: 0.2695

class 6: 0.2698

class 7: 0.2

class 8: 0.1538

class 9: 0.07143

Average specificity: 0.9187998638236754

Specificity for each class:

class 0: 0.9197

class 1: 0.9315

class 2: 0.8761

class 3: 0.9258

class 4: 0.936

class 5: 0.9477

class 6: 0.943

class 7: 0.9042

class 8: 0.8995

class 9: 0.9044

Average f1-score: 0.18530052316348514

f1-score for each class:

class 0: 0.0

class 1: 0.3833

class 2: 0.0

class 3: 0.2473

class 4: 0.3704

class 5: 0.3744

class 6: 0.3636

class 7: 0.03175

class 8: 0.05405

class 9: 0.02817

Dan berikut ini adalah hasil evaluasi dari test set:
Accuracy: 0.1775

Average sensitivity: 0.12029102624852998

Sensitivity for each class:

class 0: 0.0

class 1: 0.2667

class 2: 0.0

class 3: 0.1444

class 4: 0.137

class 5: 0.2289

class 6: 0.2239

class 7: 0.0

class 8: 0.09091

class 9: 0.1111

Average specificity: 0.909928151402488

Specificity for each class:

class 0: 0.9144

class 1: 0.9465

class 2: 0.9013

class 3: 0.9226

class 4: 0.9205

class 5: 0.9338

class 6: 0.9069

class 7: 0.8636

class 8: 0.9023

class 9: 0.8875

Average f1-score: 0.13554669737967875

f1-score for each class:

class 0: 0.0

class 1: 0.3158

class 2: 0.0

class 3: 0.2047

class 4: 0.1835

class 5: 0.3089

class 6: 0.2655

class 7: 0.0

class 8: 0.04

class 9: 0.03704

7. Buat arsitektur CNN yang baru dengan menggunakan semua hidden layer pretrained dari model sebelumnya, lakukan freezing pada semua hidden layer tersebut, lalu ganti layer output softmax dengan layer baru.

```
class network_2(nn.Module):
    def __init__(self):
        super(network_2, self).__init__()
        self.convolutional1 = nn.Conv2d(3, 25, kernel_size=3, stride=1, padding=1)
        self.maxpool1 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.relu1 = nn.ReLU()
        self.convolutional2 = nn.Conv2d(25, 50, kernel_size=3, stride=1, padding=1)
        self.maxpool2 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.relu2 = nn.ReLU()
        self.convolutional3 = nn.Conv2d(50, 100, kernel_size=3, stride=1, padding=1)
        self.maxpool3 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.relu3 = nn.ReLU()
        self.linear1 = nn.Linear(FLATTEN_SIZE_1, 100)
        self.relu4 = nn.ReLU()
        self.new_linear2 = nn.Linear(100, 10)
        self.softmax1 = nn.Softmax(dim=1)


    def forward(self, x):
        x = self.convolutional1(x)
        x = self.maxpool1(x)
        x = self.relu1(x)
        x = self.convolutional2(x)
        x = self.maxpool2(x)
        x = self.relu2(x)
        x = self.convolutional3(x)
        x = self.maxpool3(x)
        x = self.relu3(x)
        x = x.view(-1, FLATTEN_SIZE_1)
        x = self.linear1(x)
        x = self.relu4(x)
        x = self.new_linear2(x)
        x = self.softmax1(x)

        return x
```

Pada layer terakhir diganti dengan fully connected layer yang baru. Untuk layer yang lainnya menggunakan layer dari model pretrain → “model_final_4.pt”.

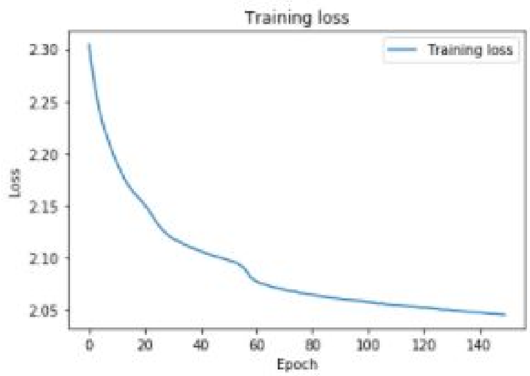
```python
In [54]: model_pretrain_2 = network_2()
         model_pretrain_2_parameter = model_pretrain_2.named_parameters()
         dict_pretrain_2 = dict(model_pretrain_2_parameter)
         load_trained = torch.load("model_final_4.pt")
         trained_dict = dict(load_trained)
         transfer_parameter(model_pretrain_2, dict_pretrain_2, trained_dict)
         freeze_all_layers(model_pretrain_2)

         #unfreeze output layer
         model_pretrain_2.new_linear2.weight.requires_grad = True
         model_pretrain_2.new_linear2.bias.requires_grad = True
```

8. Implementasikan model poin (7) pada dataset CIFAR 100 untuk kelas 1 hingga kelas 10 dengan rasio data train dan data test sebanyak 60% : 40%. Kemudian hitung matrik evaluasi (akurasi, sensitivity, specificity, dan f1-score) dari data train dan data test.

jawab:

Berikut ini adalah training loss dari model yang dilatih:



Berikut ini adalah evaluasi model pada training set:

Accuracy: 0.4242

Average sensitivity: 0.38444100478537535

Sensitivity for each class:

class 0: 0.5431

class 1: 0.5101

class 2: 0.3964

class 3: 0.2667

class 4: 0.2915

class 5: 0.4548

class 6: 0.336

class 7: 0.0

class 8: 0.4499

class 9: 0.5959

Average specificity: 0.9371660000511728

Specificity for each class:

- class 0: 0.9594
- class 1: 0.9605
- class 2: 0.9414
- class 3: 0.9005
- class 4: 0.949
- class 5: 0.9492
- class 6: 0.9416
- class 7: 0.9
- class 8: 0.9279
- class 9: 0.9421

Average f1-score: 0.3813422956316826

f1-score for each class:

- class 0: 0.5885
- class 1: 0.5739
- class 2: 0.4367
- class 3: 0.01553
- class 4: 0.391
- class 5: 0.4995
- class 6: 0.4032
- class 7: 0.0
- class 8: 0.382
- class 9: 0.523

Berikut adalah performa model pada test set:

Accuracy: 0.398

Average sensitivity: 0.33335623535050746

Sensitivity for each class:

- class 0: 0.4828
- class 1: 0.5039
- class 2: 0.3361
- class 3: 0.0
- class 4: 0.2709
- class 5: 0.4035
- class 6: 0.3433
- class 7: 0.0
- class 8: 0.4545
- class 9: 0.5385

Average specificity: 0.934217406343142

Specificity for each class:

- class 0: 0.9649
- class 1: 0.9588
- class 2: 0.9319
- class 3: 0.8997
- class 4: 0.9435
- class 5: 0.9391
- class 6: 0.9376
- class 7: 0.9
- class 8: 0.9296
- class 9: 0.9371

Average f1-score: 0.3554097604261

f1-score for each class:

class 0: 0.5714

class 1: 0.5639

class 2: 0.3653

class 3: 0.0

class 4: 0.363

class 5: 0.4299

class 6: 0.3932

class 7: 0.0

class 8: 0.3955

class 9: 0.4719

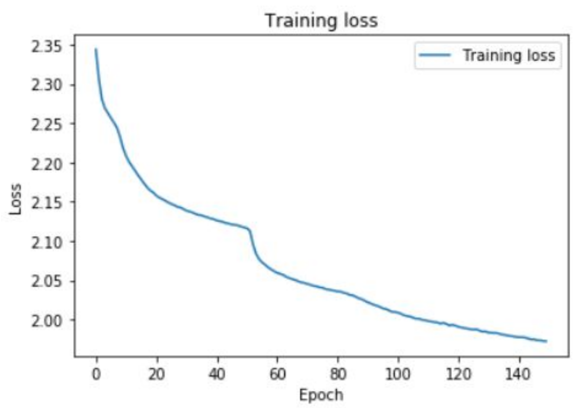
9. Lalu lakukan unfreeze pada satu hidden layer (conv-pool-actv) paling atas dan lakukan kembali proses training menggunakan deskripsi data yang sama dengan poin (8). Bagaimana hasil yang diperoleh dari perubahan ini?

```
model_pretrain_3 = network_2()
model_pretrain_3_parameter = model_pretrain_3.named_parameters()
dict_pretrain_3 = dict(model_pretrain_3_parameter)
load_trained_2 = torch.load("model_final_4.pt")
trained_dict_2 = dict(load_trained_2)
transfer_parameter(model_pretrain_3, dict_pretrain_3, trained_dict_2)
freeze_all_layers(model_pretrain_3)

#unfreeze conv layer
model_pretrain_3.convolutional3.weight.requires_grad = True
model_pretrain_3.convolutional3.bias.requires_grad = True

#unfreeze output layer
model_pretrain_3.new_linear2.weight.requires_grad = True
model_pretrain_3.new_linear2.bias.requires_grad = True
```

Berikut ini adalah loss training dari model:



Berikut ini adalah evaluasi model pada training set:
Accuracy: 0.5052

Average sensitivity: 0.41706575341274377

Sensitivity for each class:

class 0: 0.0

class 1: 0.5852

class 2: 0.4537

class 3: 0.4024

class 4: 0.0

class 5: 0.5997

class 6: 0.3581

class 7: 0.5582

class 8: 0.5032

class 9: 0.7102

Average specificity: 0.9464582695982632

Specificity for each class:

class 0: 0.9

class 1: 0.9742

class 2: 0.963

class 3: 0.9465

class 4: 0.9

class 5: 0.9671

class 6: 0.9401

class 7: 0.9517

class 8: 0.9758

class 9: 0.9462

Average f1-score: 0.44997011328074504

f1-score for each class:

- class 0: 0.0
- class 1: 0.6672
- class 2: 0.5462
- class 3: 0.4597
- class 4: 0.0
- class 5: 0.6502
- class 6: 0.4109
- class 7: 0.5621
- class 8: 0.6166
- class 9: 0.5869

Berikut ini adalah evaluasi model pada test set:
Accuracy: 0.458

Average sensitivity: 0.38590067362230274

Sensitivity for each class:

- class 0: 0.0
- class 1: 0.5396
- class 2: 0.3684
- class 3: 0.3169
- class 4: 0.0
- class 5: 0.5431
- class 6: 0.3622
- class 7: 0.4811
- class 8: 0.481
- class 9: 0.7667

Average specificity: 0.9410502499336337

Specificity for each class:

class 0: 0.9

class 1: 0.971

class 2: 0.9481

class 3: 0.9359

class 4: 0.9

class 5: 0.9581

class 6: 0.9381

class 7: 0.9452

class 8: 0.9715

class 9: 0.9426

Average f1-score: 0.40918729284875666

f1-score for each class:

class 0: 0.0

class 1: 0.6276

class 2: 0.4444

class 3: 0.3719

class 4: 0.0

class 5: 0.5833

class 6: 0.4053

class 7: 0.4951

class 8: 0.5891

class 9: 0.575

10. Tuliskan analisis anda terkait implementasi menggunakan transfer learning.

Jawab:

Untuk model transfer learning pertama yaitu fine tuning pretrain model dari soal 1-5, proses training bergerak menuju titik konvergen. Hal ini ditunjukkan oleh cross entropy loss yang terus menerus menurun selama proses training. Namun nilai cross entropy loss yang didapat seharusnya masih bisa dimaksimalkan. Penambahan jumlah epoch yang lebih besar akan meminimalkan cross entropy loss selama proses training. Proses menuju titik konvergensi yang smooth tidak lepas dari peran pretrained model yang sudah dilatih sebelumnya. Karena statistik data yang digunakan hampir mirip, maka proses adjustment pada saat melakukan fine tuning lebih mudah. Meskipun begitu, training loss pretrained model yang difinetuning pada soal 6 tidak lebih baik dibandingkan dengan model terbaik yang didapat dari soal 3. Training loss model optimal pada soal ketiga berada dikisaran 1.8 sementara training loss model pada soal nomor 6 berada di kisaran 2.01. Untuk pretrained model dari soal 7 dan 8 dilakukan freezing pada seluruh layer kecuali layer terakhir (output layer) dengan mengubahnya menggunakan layer yang baru. Selain itu, dilakukan juga perubahan untuk training data dan test data. Dari dataset CIFAR100, diambil hanya data yang memiliki kelas 1 sampai 10 saja (indeks 0 - 9 bila menggunakan framework torchvision). Berdasarkan evaluasi metrik (accuracy, specificity, sensitivity, dan f1 score) pada test set dan training set, Model pretrain yang kedua memiliki nilai akurasi yang lebih baik dibandingkan dengan model pretrain pertama (soal 6). Model pretrain kedua memiliki nilai rata-rata sensitivitas yang lebih tinggi dibandingkan dengan model pretrain pertama. Model kedua juga memiliki rata-rata spesifisitas yang lebih tinggi dibandingkan model pretrain pertama. Model pretrain kedua (soal 7 dan 8) juga memiliki nilai rata-rata f1 score yang lebih tinggi dibandingkan model pertama. Secara umum, model pretrain kedua memiliki performa yang lebih baik dibandingkan dengan model pretrain pertama. Meskipun seluruh layer pada model pretrain pertama di-unfreeze, namun jumlah data yang kurang banyak menyebabkan performa model pretrain kedua lebih baik dibandingkan dengan model pretrain yang pertama. Pada model pretrain kedua, performanya ditentukan oleh kemampuan tuning dari layer terakhir untuk meminimalkan cross entropy loss. Selanjutnya untuk model pretrain ketiga (soal 9), metrik akurasi, sensitivity, specificity, dan f1 score dari training set dan test set memberi hasil yang lebih tinggi dibandingkan dengan model pretrain kedua. Hal ini diduga karena lebih banyak layer yang dapat dituning. Terlebih lagi dimungkinkannya tuning pada layer terakhir convolutional layer memungkinkan model untuk memaksimalkan ekstraksi high level feature. Adanya high level feature dapat membantu model untuk mengidentifikasi kelas dari model. Dari sisi training, model pretrain ketiga lebih mendekati titik konvergen dibandingkan dengan dua model pretrain lainnya. Hal ini ditunjukkan oleh grafik loss function ketiga model dimana model pretrain ketiga lebih mendekati nilai minimum cross entropy loss (bernilai 0) dibandingkan kedua model lainnya. Secara umum, ketiga model teroptimisasi dengan secara smooth. Hanya saja, jumlah epoch yang kurang besar menyebabkan performa ketiga model pretrain kurang maksimal. Kita dapat menyimpulkan bahwa model yang dilatih pada dataset yang memiliki karakteristik yang mirip dengan dataset saat ini dapat membantu meningkatkan performa model dalam melakukan klasifikasi data.