Laporan Proyek Machine Learning
“Human Detection dengan metode CNN”

Bryan Elmer Cahyadi / C14190160
Verrel Ravanelli Santoso / C14190152
Marselus Richard Gianto / C14190120

Penjelasan metode & *dataset*

Pada proyek mata kuliah *machine learning* guna memenuhi nilai akhir dari matakuliah ini, kami memilih topik mencoba membuat *human detection with CNN (Convolutional Neural Network) method*. Pada proyek ini, kami menggunakan 2 file dengan bahasa pemrograman *python* yaitu train.py dan test.py. Untuk menguji dataset yang telah kami siapkan terdapat 2 tahap untuk menjalankan program. Tahapan pertama pengguna perlu menjalankan file train.py lalu tahap kedua perlu menjalankan test.py. *File* train.py berguna untuk melakukan *training* pada *dataset* yang telah disiapkan untuk menghasilkan *model* yang nantinya akan berguna untuk tahap kedua yaitu saat *file* test.py dijalankan untuk menguji inputan gambar yang disiapkan oleh user.

Dataset yang disiapkan berupa foto-foto sejumlah 4770 foto. Kumpulan foto tersebut dibagi kedalam 2 *folder* yaitu *folder neg* dan *folder pos*. *Folder neg* berisikan foto-foto yang tidak terdapat objek manusia di dalamnya sedangkan untuk *folder pos* berisikan foto-foto yang terdapat objek manusia di dalamnya. Pada tahap yang pertama *folder neg* berisikan 2772 foto dan *folder pos* berisikan 1998 foto.

Dalam proyek kali ini kami mencoba untuk melakukan percobaan sebanyak 4 tahap sesuai dengan tahapan dalam pemisahan data set menggunakan metode CNN yang tidak kami lakukan perubahan dan tambahan 2 kali dengan mencoba merubah susunan *layer* dan argumen pada metode CNN yang kami gunakan. 4 tahapan perubahan dataset terdiri dari tahapan pertama yaitu foto yang mengandung objek berbentuk patung perlu dipindahkan ke dalam *folder neg*. Tahapan kedua yaitu foto dengan objek manusia yang berkerumunan sampai tidak terlihat jelas lagi objek manusianya perlu dipindahkan ke dalam *folder neg*. Tahapan ketiga yaitu foto dengan objek manusia yang berkerumunan tetapi jelas terlihat objek manusianya dipindahkan ke *folder neg*. Tahapan keempat yaitu hanya menyisakan foto dengan objek 1 orang manusia pada *folder pos*.

Dokumentasi Kode Program

- **Train.py**

```
# -*- coding: utf-8 -*-

""" Convolutional network applied to CIFAR-10 dataset classification
task.
References:
    Learning Multiple Layers of Features from Tiny Images, A.
Krizhevsky, 2009.
Links:
    [CIFAR-10 Dataset](https://www.cs.toronto.edu/~kriz/cifar.html)

    CSAL4243 Introduction to Machine Learning's assignment 3.
```

```
"""
from __future__ import division, print_function, absolute_import

import tflearn
from tflearn.data_utils import shuffle, to_categorical
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.estimator import regression
from tflearn.data_preprocessing import ImagePreprocessing
from tflearn.data_augmentation import ImageAugmentation
from tflearn.layers.normalization import
local_response_normalization
from tflearn.data_utils import image_preloader

# Data loading and preprocessing

data_dir = "Data"
model_path = "vggmodel"
# the file gen by generated by gen_files_list.py
#files_list = "/path/to/your/file/with/images""

#or use the mode 'Data'
X, Y = image_preloader(data_dir, image_shape=(128, 128),
mode='folder',categorical_labels=True, normalize=True,
files_extension=['.jpg', '.png'], filter_channel=True)

# Real-time data preprocessing
img_prep = ImagePreprocessing()
img_prep.add_featurewise_zero_center()
img_prep.add_featurewise_stdnorm()

# Real-time data augmentation
img_aug = ImageAugmentation()
img_aug.add_random_flip_leftright()
img_aug.add_random_rotation(max_angle=25.)
img_aug.add_random_blur(sigma_max=5.)

# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
```

```
network = conv_2d(network, 64, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 32, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 32, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 256, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2, activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)

# Train using classifier
model = tflearn.DNN(network, tensorboard_verbose=0)

model.fit(X, Y, n_epoch=40, validation_set=0.1, shuffle=True,
          show_metric=True, batch_size=10, snapshot_epoch=False,
          snapshot_step=20, run_id='imlprojectss')

model.save('model/newmodel.tfl')
```

- **Test.py**

```
# -*- coding: utf-8 -*-

""" Convolutional network applied to CIFAR-10 dataset classification
task.
References:
    Learning Multiple Layers of Features from Tiny Images, A.
Krizhevsky, 2009.
Links:
    [CIFAR-10 Dataset](https://www.cs.toronto.edu/~kriz/cifar.html)

    CSAL4243 Introduction to Machine Learning's assignment 3.
"""
from __future__ import division, print_function, absolute_import

import tflearn
from tflearn.data_utils import shuffle, to_categorical
```

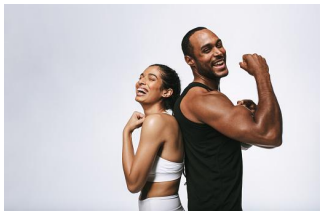
```
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.estimator import regression
from tflearn.data_preprocessing import ImagePreprocessing
from tflearn.data_augmentation import ImageAugmentation
from tflearn.layers.normalization import
local_response_normalization
from tflearn.data_utils import image_preloader
import numpy as np
# Data loading and preprocessing

data_dir = "tData"
model_path = "vggmodel"
# the file gen by generated by gen_files_list.py
#files_list = "/path/to/your/file/with/images""

#or use the mode 'Data'
X, Y = image_preloader(data_dir, image_shape=(128,128),
mode='folder', normalize=True, categorical_labels=True,
files_extension=['.jpg', '.png'], filter_channel=True)

# Real-time data preprocessing
img_prep = ImagePreprocessing()
img_prep.add_featurewise_zero_center()
img_prep.add_featurewise_stdnorm()

# Real-time data augmentation
img_aug = ImageAugmentation()
img_aug.add_random_flip_leftright()
img_aug.add_random_rotation(max_angle=25.)
img_aug.add_random_blur(sigma_max=5.)

# Convolutional network building
network = input_data(shape=[None, 128, 128, 3],
data_preprocessing=img_prep, data_augmentation=img_aug)
network = conv_2d(network, 64, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 32, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 32, 3, activation='relu')
```

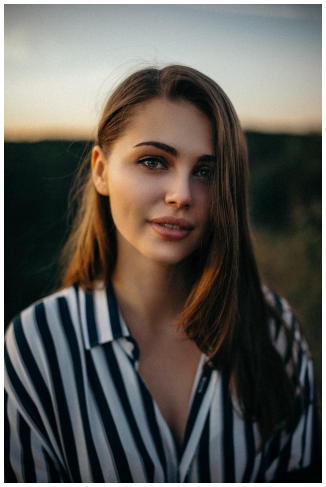
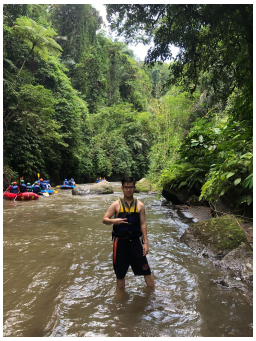
```
network = max_pool_2d(network, 2)
network = fully_connected(network, 256, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2, activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)

# Train using classifier
model = tflearn.DNN(network, tensorboard_verbose=0)
model.load('model/newmodel.tfl')

# predict test images label
X = np.array(X)
#print(X)
Y = np.array(Y)
y_pred = model.predict(X)
print(y_pred)
print(np.argmax(Y, axis=1))

# Compute accuracy of trained model on test images
print ("Accuracy: ",np.sum(np.argmax(y_pred, axis=1) == np.argmax(Y,
axis=1))*100/Y.shape[0],"%")
print(np.argmax(y_pred, axis=1) == np.argmax(Y, axis=1))
```

Hasil dan Analisa

- **Inputan :**

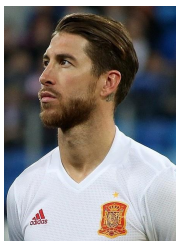
  Inputan Test berupa 30 gambar di luar dataset yaitu,
  15 Manusia :





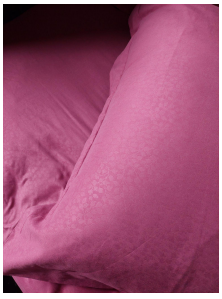



15 Non-manusia :

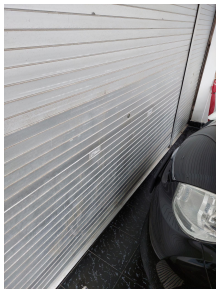








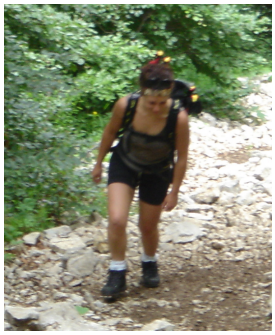
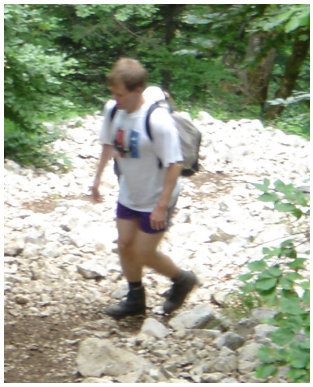
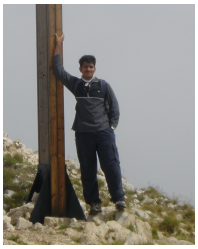
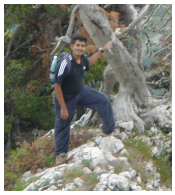




Inputan Test lainnya berupa 30 gambar di dalam dataset yaitu,
15 Manusia :










15 Non-manusia :





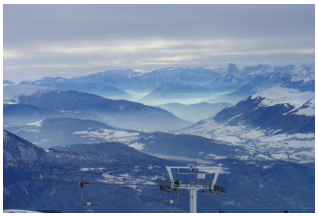












- Percobaan dengan merubah jumlah epoch dengan dataset asli :

Perbandingan dengan modifikasi jumlah epoch		
Epoch	Akurasi - inputan gambar di luar dataset	Akurasi - inputan gambar dari dataset
5	62.06	83
10	75.86	80
20	72.41	86.66
40	68.96	93.33
80	62.06	90

- Percobaan dengan pemilahan dataset dengan jumlah epoch 10:
  - Hasil :

Tahapan pemilahan *dataset*	Akurasi
Tahapan 1	72.41
Tahapan 2	62.06
Tahapan 3	51.72
Tahapan 4	51.72

- **Percobaan memodifikasi susunan *layer* dan angka pada parameter dan *dataset* asli serta epoch 10** :
  - Percobaan 1
    - Layer :
      - Parameter Layer : 128, 64, 64

```
# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = max_pool_2d(network, 2)
network = conv_2d(network, 64, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 64, 3, activation='relu')
```

```
network = max_pool_2d(network, 2)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2,
activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)
```

- Hasil : 68.96%

- Percobaan 2
  - Layer :
    - Parameter Layer: 256, 256, 128, 128, 256

```
# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
```

```
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2,
activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)
```

- Hasil : 65.5%

- Percobaan 3
  - Layer :
    - Parameter Layer: 512, 256, 128

```
# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
network = conv_2d(network, 512, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2,
activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)
```

- Hasil : 65.51%

- Percobaan 4
  - Layer :
    - Parameter Layer : 1024, 64, 128

```
# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
network = conv_2d(network, 1024, 3, activation='relu')
```

```
network = max_pool_2d(network, 2)
network = conv_2d(network, 64, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2,
activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)
```

- Hasil : 72.41%

- Percobaan 5
  - Layer :
    - Parameter Layer: 4, 2, 16

```
# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
network = conv_2d(network, 4, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 2, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 16, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2,
activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)
```

- Hasil : 62.06%

- Percobaan 6
  - Layer :
    - Parameter Layer: 32, 8, 2048

```
# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
network = conv_2d(network, 32, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 8, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 2048, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2,
activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)
```

  - Hasil : 58.62%

- Percobaan 7
  - Layer :
    - Parameter Layer: 256, 8 ,128,16

```
# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 8, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 16, 3, activation='relu')
network = max_pool_2d(network, 2)
```

```
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2,
activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)
```

- Hasil : 65.51%

- Percobaan 8
  - Layer :
    - Parameter Layer: 512, 64, 32, 128, 8, 4, 512, 1024, 256, 512, 32, 32

```
# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
network = conv_2d(network, 512, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 64, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 32, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 8, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 4, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 512, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 1024, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 512, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 32, 3, activation='relu')
```

```
network = max_pool_2d(network, 2)
network = conv_2d(network, 32, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 512,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 1024,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 128,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 64, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 8, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2,
activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)
```

        - Hasil : 62.06%
    - Percobaan 9
        - Layer :
            - Parameter Layer: 128, 128, 128, 128, 128

```
# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 64, activation='relu')
network = dropout(network, 0.5)
```

```
network = fully_connected(network, 64, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 32, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 32, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2,
activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)
```

- Hasil : 62.06%

- Percobaan 10
  - Layer :
    - Parameter Layer: 256, 128, 32, 8, 4

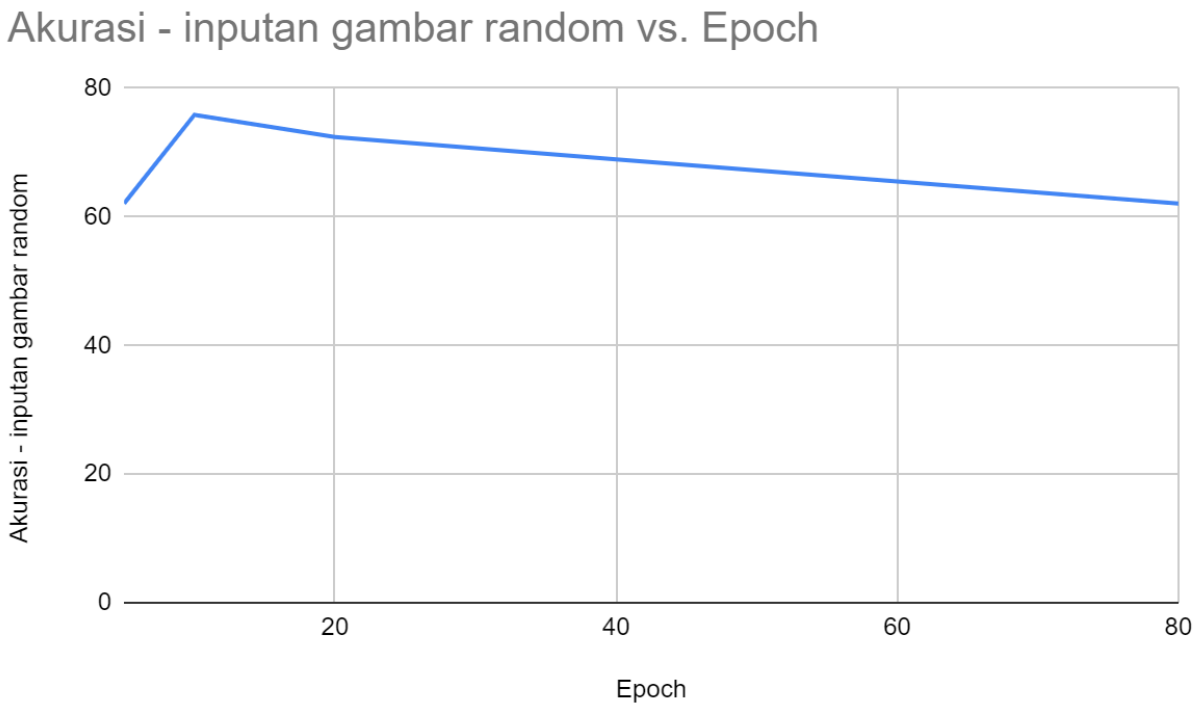
```
# Convolutional network building
network = input_data(shape=[None, 128, 128,
3],data_preprocessing=img_prep,data_augmentation=img_aug)
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 32, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 8, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 4, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 512,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 256,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 128,
activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 64, activation='relu')
network = dropout(network, 0.5)
```

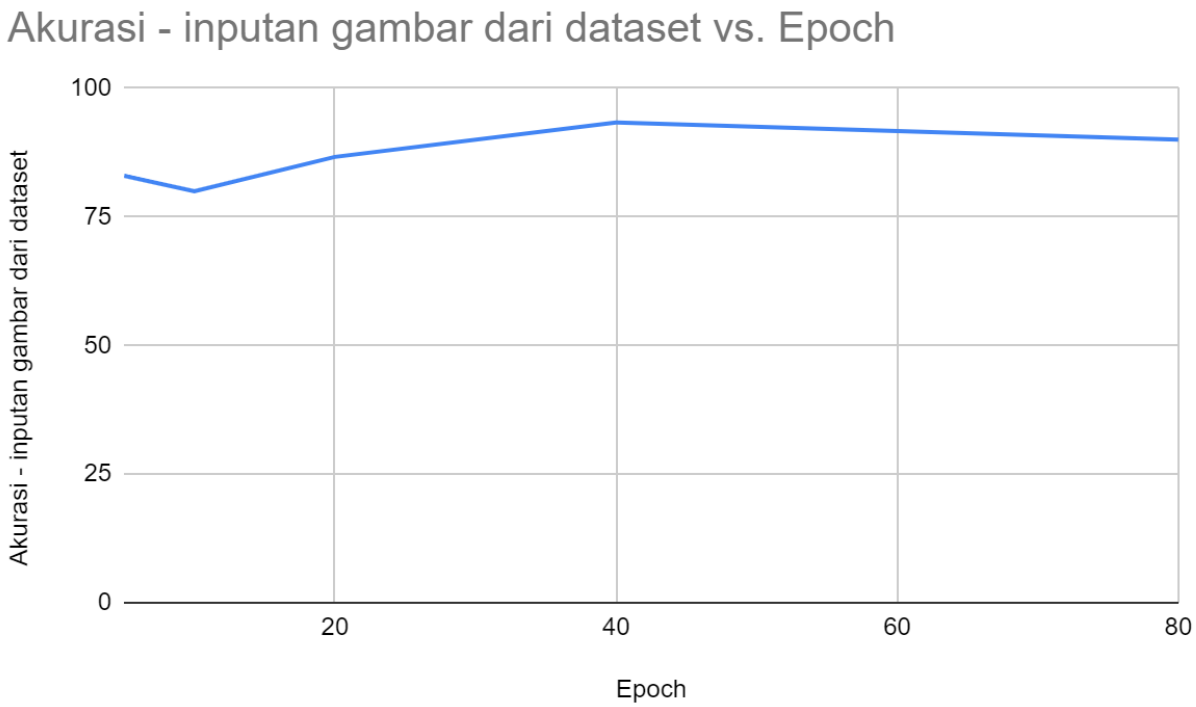
```
network = fully_connected(network, 2,
activation='softmax')
network = regression(network, optimizer='adam',
loss='categorical_crossentropy', learning_rate=0.001)
```

- Hasil : 51.72%

Kesimpulan

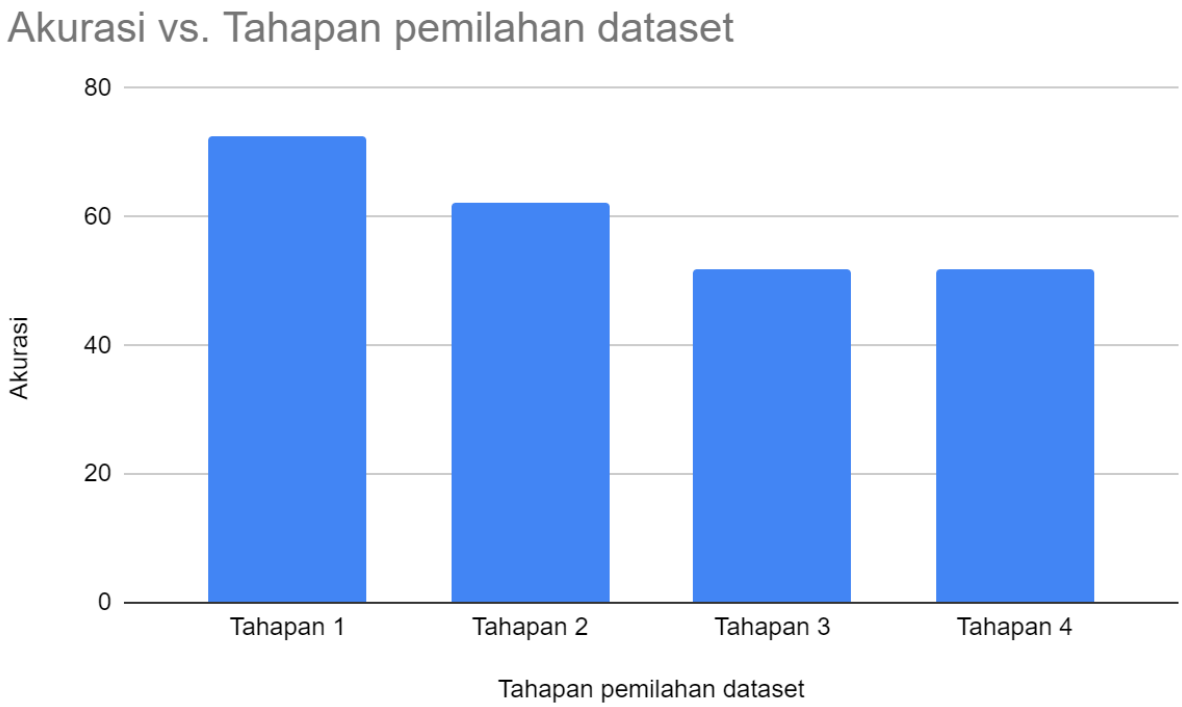
- **Perubahan epoch**





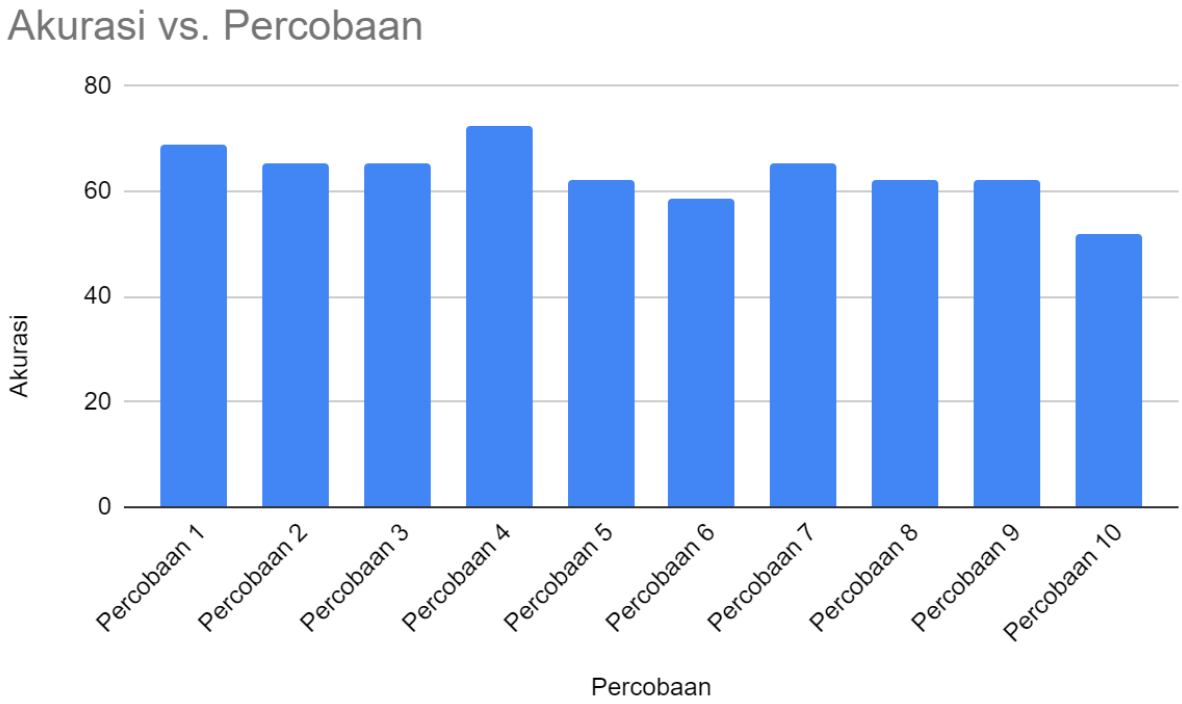
Dapat dilihat dari perbandingan akurasi yang dihasilkan, bahwa jumlah epoch yang lebih banyak belum tentu menghasilkan akurasi yang lebih baik. Memang benar bahwa terdapat kecenderungan peningkatan akurasi. Tetapi dapat dilihat dari tabel di atas ketika input gambar yang dimasukan merupakan gambar-gambar di luar dataset maka terjadi penurunan akurasi dari epoch 5 ke 10 dan juga dapat dilihat juga ketika input gambar merupakan gambar dari dataset maka terjadi penurunan akurasi dari epoch 5 ke 10. Inputan gambar sangat menentukan besar dari akurasi yang dihasilkan oleh proyek ini.

- **Pemilahan *dataset***



Grafik berikut menunjukan perubahan akurasi yang terjadi pada saat dilakukannya pemilahan dataset. Pengujian di atas dilakukan dengan menggunakan 10 epoch dan inputan berupa foto random di luar dataset. Tahapan 1 merupakan pemilahan dataset memisahkan foto dengan objek ambigu atau menyerupai manusia pada folder pos yang berisi seluruh foto manusia ke dalam folder neg yang berisi foto non manusia. Dapat dilihat terjadi penurunan dibandingkan dengan *training* dengan dataset asli. Dengan dataset asli dan epoch 10 kali menunjukan akurasi 75.86% sedangkan dataset pada tahap 1 dengan epoch 10 kali menyatakan akurasi 72.41%. Pada tahapan 2 terjadi penurunan akurasi yang dapat dilihat pada grafik batang di atas. Tahapan 2 merupakan tahapan pemilahan foto-foto dengan objek manusia yang berkerumunan dan terdapat banyak manusia yang tidak terlihat jelas pada foto dipindahkan kedalam folder neg. Pada tahap 2 ini tercatat akurasi hanya sebesar 62.06% Lalu akurasi semakin berkurang jauh pada saat memasuki pada pemilahan dataset tahap ke 3. Tahap 3 merupakan tahapan di mana memilah foto-foto dengan objek manusia yang terdapat beberapa manusia di dalamnya ke dalam folder neg. Pada tahapan 3 tercatat penurunan akurasi yang semakin jauh menjai 51.72% saja. Dan pada tahapan terakhir yaitu tahapan 4 dimana folder pos hanya menyisakan foto-foto dengan 1 objek manusia saja. Pada tahapan 4 ini tidak terdapat perubahan akurasi dibandingkan dengan tahap 3.

- **Modifikasi susunan layer**



Grafik di atas menunjukan perubahan akurasi yang berhasil dicatat pada saat layer metode CNN dimodifikasi. Tidak terdapat perubahan yang signifikan antara percobaan pertama dengan percobaan kedua. Namun, akurasi dengan layer yang tidak dimodifikasi mengalami penurunan. Percobaan dengan 10 epoch. layer CNN yang tidak dimodifikasi , training dataset asli tanpa di pilihan, dan input berupa foto-foto random di luar dataset mencatatkan akurasi sebesar 75.86% sedangkan percobaan 1 menghasilkan akurasi sebesar 68.96% dan percobaan 2 sebesar 65.5%.

- **Kesimpulan umum**

  Dilihat dari analisis dan pengamatan kelompok kami, akurasi yang dihasilkan dipengaruhi oleh banyak hal. Mulai dari *layer CNN*, inputan foto, *dataset*, dan epoch. Semakin banyak epoch tidak menunjukan akan menghasilkan akurasi yang semakin baik. Dalam tahapan modifikasi *layer CNN* terdapat kemungkinan bahwa kelompok kami belum dapat meramu *layer* dan parameter yang tepat untuk menghasilkan akurasi yang lebih baik. Pada tahapan pemisahan *dataset* menurut pendapat kelompok kami berdasarkan hasil dari observasi. Dapat kami simpulkan bahwa apabila pada tahapan folder neg berisi foto-foto manusia maka akan terjadi ambiguitas yang mana menyebabkan penurunan nilai akurasi tersebut.