

# Deteksi Penggunaan Masker Wajah

## Menggunakan YOLO



Disusun oleh:

August Berlin T (C14180003)

Eric Saputra L (C14180048)

Steven A S (C14180049)

Kevin Giovanni (C14180086)

Vincent Natanael (c14180141)

## Latar Belakang

Pandemi virus SARS-Cov-II yang menyebabkan penyakit COVID-19 adalah pandemi pertama yang muncul di era modern. Dengan tingkat pergerakan manusia yang tertinggi selama sejarah manusia karena perkembangan infrastruktur dan transportasi, persebaran virus ini menjadi sangat cepat dikarenakan banyaknya manusia yang bertemu satu dengan yang lainnya. Virus ini telah menjangkit dan membunuh jutaan jiwa dan persebarannya akan menjadi tidak terkendali apabila laju persebarannya tidak ditekan.

Dalam rangka menghambat persebaran virus tersebut, pemerintah telah mengupayakan berbagai cara, dari vaksinasi hingga mengkampanyekan masyarakat untuk memakai masker, menjaga jarak, dan mencuci tangan. Namun usaha-usaha tersebut memerlukan pengawasan yang ketat untuk memastikan bahwa masyarakat memang telah mematuhi protokol kesehatan. Bagi para pelanggar, perlu diberikan pembinaan dan/atau tindakan.

Deteksi penggunaan masker menggunakan YOLO diharapkan dapat menjadi salah satu cara untuk mempermudah dan meningkatkan upaya pengawasan masyarakat agar patuh pada protokol kesehatan, khususnya dalam poin mengenakan masker ketika berada di luar rumah.

# Cara Kerja

## 2.1. YOLO v1 (Redmon, Divvala, Girshick, & Farhadi, 2016)

Metode *You Only Look Once* atau disingkat YOLO bertujuan untuk melakukan *object classification* dengan cepat dan ringan dengan penurunan akurasi yang relatif rendah. Kecepatan ini dicapai dengan menghindari iterasi berulang untuk mendekripsi setiap class dalam image. Oleh karena itu, YOLO mencoba melakukan deteksi untuk berbagai class sekaligus.

Hal ini dilakukan dengan terlebih dahulu memecah *image* menjadi beberapa bagian dengan ukuran masing-masing baginya sama. Untuk setiap bagian, dihitung probabilitas bagian tersebut adalah sebuah *object*. Hal ini nantinya digunakan dalam menentukan posisi *bounding box* pada *object detection*.

Pada saat yang bersamaan, untuk setiap bagian, model juga langsung menghitung *object* apa yang ada dalam bagian tersebut jika diasumsikan itu adalah *object*. Dengan demikian, model dapat langsung mengetahui klasifikasi *object* sebelum mengetahui posisi *object* sehingga prosesnya dapat dilakukan dengan lebih cepat. Namun, memprediksi *object* sebelum mengetahui adanya *object* akan menghasilkan banyak deteksi yang tidak relevan. Oleh karena itu, *confidence* juga dihitung untuk setiap klasifikasi dan menerapkan *threshold confidence* minimal sebagai salah satu syarat agar klasifikasi tersebut dipakai dalam *object detection* nantinya.

## 2.2. YOLO v2 (Redmon & Farhadi, 2016)

YOLO v2 berusaha mengembangkan mekanisme YOLO v1. YOLO v2 menggunakan resolusi *image* yang sama pada waktu *training* dan waktu digunakan sehingga fitur-fitur yang diperhatikan lebih mendetail. Sebagai perbandingan, YOLO v1 mengurangi resolusi pada

waktu *training* sehingga mempercepat proses *training* namun mengurangi akurasi.

Dalam memprediksi posisi *bounding box*, YOLO v2 memprediksi *anchor box* menggunakan *K-means clustering*, lalu memprediksi *bounding box* yang sebenarnya sebagai *offset* dari *anchor box* tersebut. Prediksi *offset* umumnya lebih cepat daripada prediksi koordinat sehingga penentuan posisi *bounding box* dapat dipercepat.

Beberapa perbedaan arsitektur lainnya seperti jumlah *layer* juga meningkatkan akurasi YOLO v2 dibandingkan dengan YOLO v1.

### 2.3. YOLO v3 (Redmon & Farhadi, 2018)

YOLO v3 melakukan pengembangan dan percobaan metode baru yang bertujuan untuk meningkatkan efektivitas YOLO sambil menjaga tingkat akurasi. YOLO v3 menggunakan *logical regression* untuk melakukan prediksi dalam pembuatan *bounding box*, berbeda dengan YOLO v2 yang memakai *anchor boxes* yang meningkatkan kecepatan dalam memprediksi lokasi *bounding box*. YOLO v3 juga mengganti metode *class prediction* karena *bounding box* dapat mengandung beberapa klasifikasi, seperti sebuah foto perempuan memiliki klasifikasi perempuan dan orang, menjadi *independent logistic classifiers* dikarenakan *softmax* tidak berpengaruh terhadap performa YOLO sehingga YOLO v3 memakai *independent logistic classifiers*.

Untuk melakukan *feature extraction* YOLO v3 memakai versi modifikasi dari Darknet-19 yang digunakan oleh YOLO v2, yang disebut Darknet-53. Darknet-53 memiliki 53 *convolutional layers* yang lebih kuat daripada Darknet-19 tetapi tetap menjaga kecepatan dan efisiensi.

YOLO v3 kesulitan mendeteksi benda yang berukuran sedang dan besar. Hal ini berbanding terbalik dengan YOLO versi sebelumnya yang kesulitan mendeteksi benda berukuran kecil.

## 2.4. YOLO v4 (Bochkovskiy, Wang, & Liao, 2020)

YOLO v4 mempertahankan struktur dan cara kerja yang sama dengan YOLO v3, namun dengan jumlah layer, kedalaman layer, dan resolusi yang lebih besar. Fungsi-fungsi yang digunakan dalam YOLO v3 diganti dengan fungsi-fungsi lain yang telah diteliti lebih lanjut dan ditemukan lebih baik, setidaknya dalam kasus di mana YOLO digunakan. Sebagai contoh, *activation function sigmoid* diganti dengan DropBlock.

Selain perubahan fungsi yang digunakan, YOLO v4 menggunakan strategi *training* yang berbeda tanpa meningkatkan waktu deteksi. Dua hingga empat *image* untuk *training* dicampur menggunakan *CutMix* dan *Mosaic* untuk mengurangi jumlah data *training* yang diperlukan. Selain itu, *image* asli diubah dalam *Self-Adversarial Training* lalu digunakan sebagai input untuk *training*.

Perubahan-perubahan di atas meningkatkan kecepatan dan akurasi YOLO v4 terhadap YOLO v3.

# Training dan Hasil

## 3.1. Detail Training

*Training* dilakukan menggunakan *framework* Darknet yang dikembangkan oleh Alexey (<https://github.com/AlexeyAB/darknet>). Untuk melakukan *training, file-file* berikut disiapkan:

- obj.names mengandung nama-nama kelas yang akan di train
- obj.data mengandung informasi jumlah class, lokasi path dari file train.txt dan test.txt, file \*.names, dan folder tempat penyimpanan weight. Jika path yang dimasukkan adalah relative path, path akan relatif terhadap executable darknet.
- train.txt mengandung daftar path gambar-gambar yang digunakan untuk *training* relatif terhadap *executable* darknet
- test.txt mengandung daftar gambar-gambar yang digunakan untuk *testing* relatif terhadap *executable* darknet
- *File-file image* yang diberi *annotation labels* dalam file \*.txt dengan nama yang sama dengan *file image* tersebut

```
darknet
| -cfg
|   |-network-settings.cfg
| -data
|   |-obj
|   |   |-obj.names
|   |   |-obj.data
|   |-test
|   |   |-train.txt
|   |   |-test.txt
```

Untuk persiapan *image file* untuk *training* dan *testing*, setiap *image* diberi *text file* dengan nama yang sama dengan nama *image file* tersebut. *Text file* tersebut berisi anotasi dalam format berikut:

```
# className center-x center-y width height  
0 0.479375 0.49130434782608695 0.85375  
0.9721739130434782
```

Setiap *bounding box* menggunakan satu baris. Variabel *className* menunjukkan kelas yang ditunjuk oleh *bounding box*. Variabel *center-x* dan *center-y* menunjukkan koordinat titik pusat dari *bounding box*. Variable *width* dan *height* menunjukkan ukuran dari *bounding box*. Nilai dari keempat koordinat tersebut dihitung relatif terhadap ukuran *image* dengan range di atas 0 dan tidak melebihi 1.0.

Setelah *file-file* disiapkan, *training* dilakukan dengan *command* berikut:

```
./darknet detector train <path ke obj.data> <path ke  
custom config> <path ke weight file>
```

*Training* dilakukan menggunakan model Yolo v3 tiny dan Yolo v4 tiny karena waktu yang diperlukan relatif cepat terhadap model-model lainnya. Selain itu, tidak semua komputer memiliki *hardware* yang relatif kuat sehingga model-model yang lebih kecil diperlukan.

### 3.2. Dataset yang Digunakan

Dataset yang digunakan untuk training diambil dari Face Mask Dataset (YOLO Format) oleh Aditya Purohit (link: <https://www.kaggle.com/aditya276/face-mask-dataset-yolo-format>).

Dataset ini terdiri atas tiga folder:

- Train folder: 610 *jpg files*, 82 *jpeg files*, 8 *png files*
- Test folder: 105 *jpg files*, 12 *jpeg files*, 3 *png files*
- Validation folder (tidak digunakan dalam percobaan ini): 100 *image files*

Dalam dataset tersebut, ada dua label: unmasked dan masked. Foto-foto dalam dataset tersebut dominan mengandung yang menggunakan masker.

### 3.3. Penjelasan Eksperimen

- Eksperimen pertama dilakukan untuk mengukur tingkat akurasi model Yolo V3 tiny dalam mendeteksi penggunaan masker dengan dataset dari Kaggle.
- Eksperimen kedua dilakukan untuk mengukur pengaruh peningkatan variasi dataset terhadap akurasi model YOLO v3 tiny.
- Eksperimen ketiga dilakukan untuk mengukur pengaruh *preprocessing* data *training* dan *testing* terhadap tingkat akurasi model YOLO v3 tiny.
- Eksperimen keempat dilakukan untuk mengukur pengaruh perubahan *learning rate* terhadap akurasi model YOLO v3 tiny. Dalam eksperimen keempat, nilai *learning rate* digandakan terhadap nilai learning rate dari eksperimen kedua.
- Eksperimen kelima dilakukan untuk mengukur peningkatan akurasi YOLO v4 terhadap YOLO v3.
- Seperti eksperimen ketiga, eksperimen keenam dilakukan untuk mengukur pengaruh *preprocessing* data *training* dan *testing* terhadap tingkat akurasi model YOLO v4 tiny.

### 3.4. Settings untuk Semua Eksperimen

- Build flags (makefile):
  - CUDNN\_HALF = 1
  - CUDNN = 1 (Digunakan agar Darknet dapat menggunakan CUDNN library untuk mempercepat *training* dan *inference* karena Google Colab menggunakan GPU NVIDIA)

- GPU = 1 (Digunakan agar Darknet dapat menggunakan GPU)
- OPENCV = 1
- Cfg settings:
  - Jumlah class = 2 class
  - Jumlah maksimum step = 6000 steps
    - Tidak kurang dari jumlah *image* untuk *training* = di atas 700
    - Tidak kurang dari jumlah class \* 2000 = di atas 4000
    - Tidak kurang dari 6000 = di atas 6000
  - line\_steps = 4800, 5400
    - Nilai yang digunakan adalah 80% dan 90% dari jumlah langkah maksimum (6000)
  - Batch size = 64
  - Subdivision = 16
  - Resolusi = 416 x 416
- Platform: Google Colab
  - GPU: NVIDIA T4
  - Versi CUDA: 11.0
  - Versi CUDNN: 7.6.5

### 3.5. Settings per Eksperimen

- Eksperimen Pertama

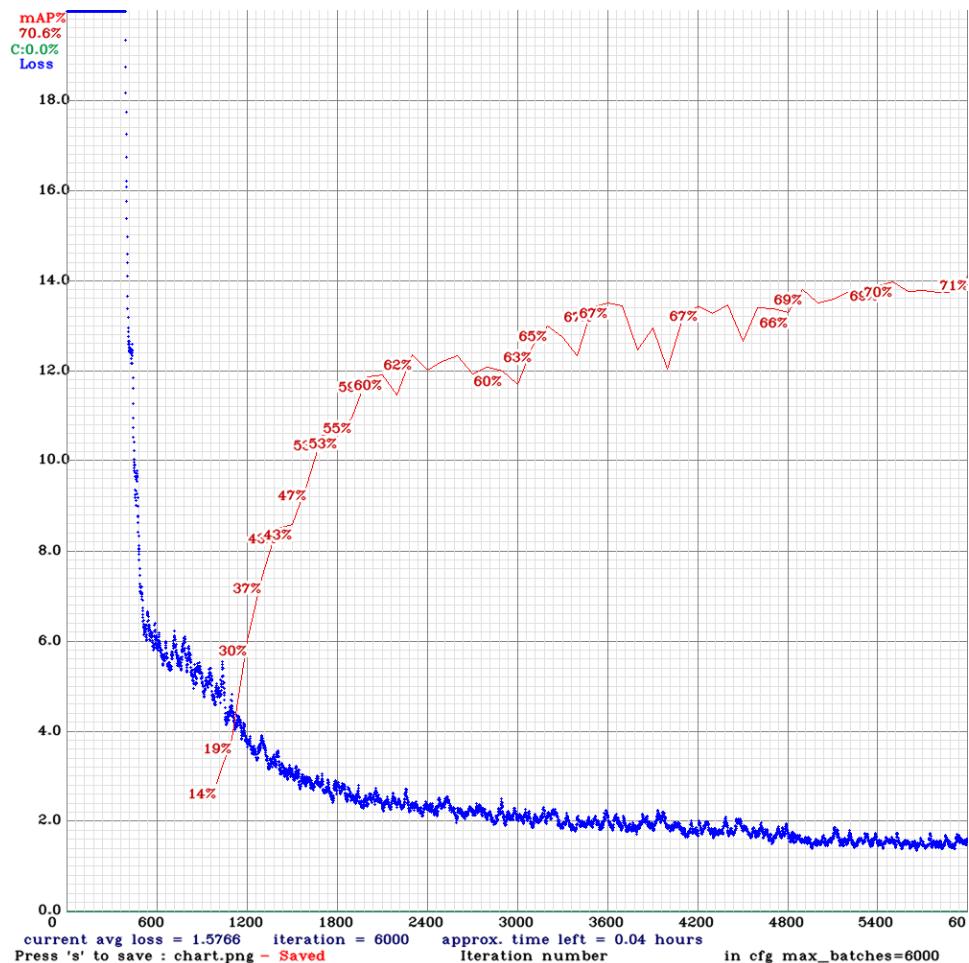
- Model: Yolo v3 tiny
- Dataset yang digunakan: 105 untuk testing, 610 untuk training
- Learning rate: 0.001
- Eksperimen Kedua
  - Model: Yolo v3 tiny
  - Dataset yang digunakan: 120 untuk testing, 700 untuk training
  - Learning rate: 0.001
- Eksperimen Ketiga
  - Model: Yolo v3 tiny
  - Dataset yang digunakan: 120 untuk testing dengan *histogram equalization*, 700 untuk training dengan *histogram equalization*
  - Learning rate: 0.001
- Eksperimen Keempat
  - Model: Yolo v3 tiny
  - Dataset yang digunakan: 120 untuk testing, 700 untuk training
  - Learning rate: 0.002
- Eksperimen Kelima
  - Dataset yang digunakan: 120 untuk testing, 700 untuk training
  - Model: Yolo v4 tiny

- Learning rate: 0.002
- Eksperimen Keenam
  - Dataset yang digunakan: 120 untuk testing dengan *histogram equalization*, 700 untuk training dengan *histogram equalization*
  - Model: Yolo v4 tiny
  - Learning rate: 0.002
- Eksperimen Ketujuh
- Eksperimen Kedelapan

# Hasil Eksperimen

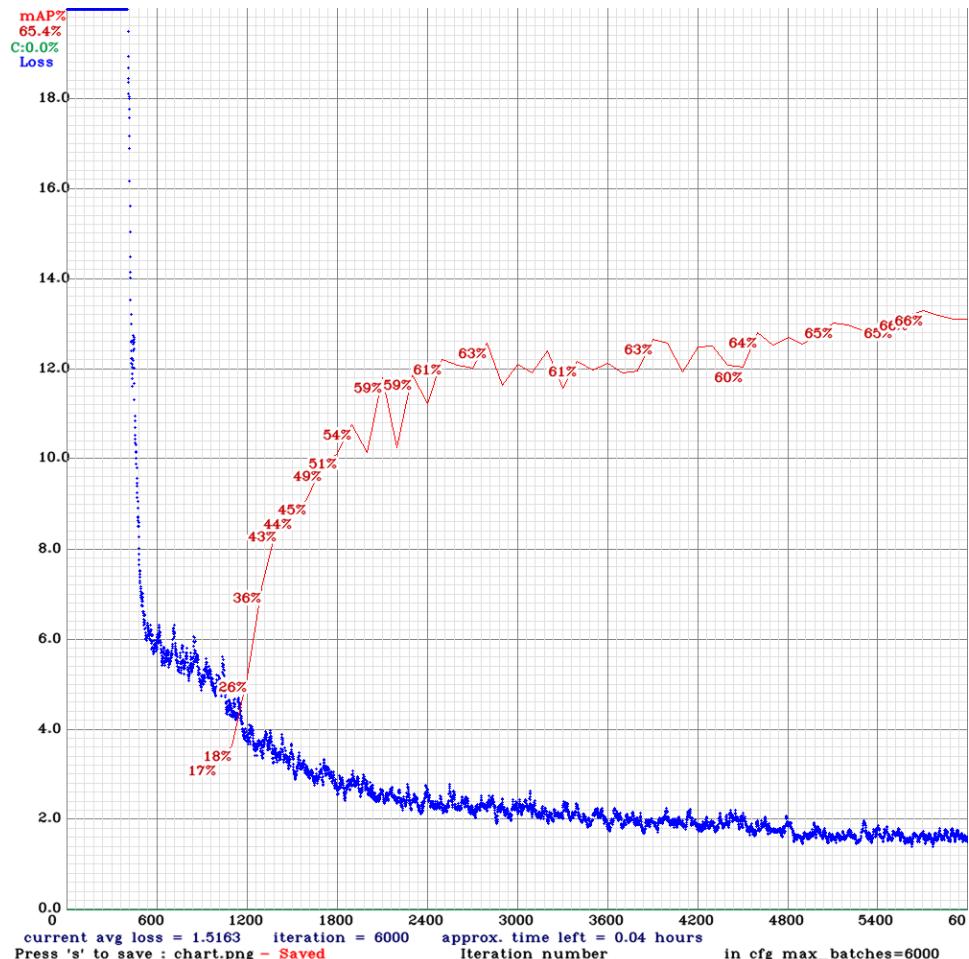
## 4.1. Eksperimen Pertama

- Lama training: 3 jam 35 menit
- Learning rate: 0.001
- mAP: 70.6%
- no\_mask ap = 47.04%
- mask ap = 82.00%



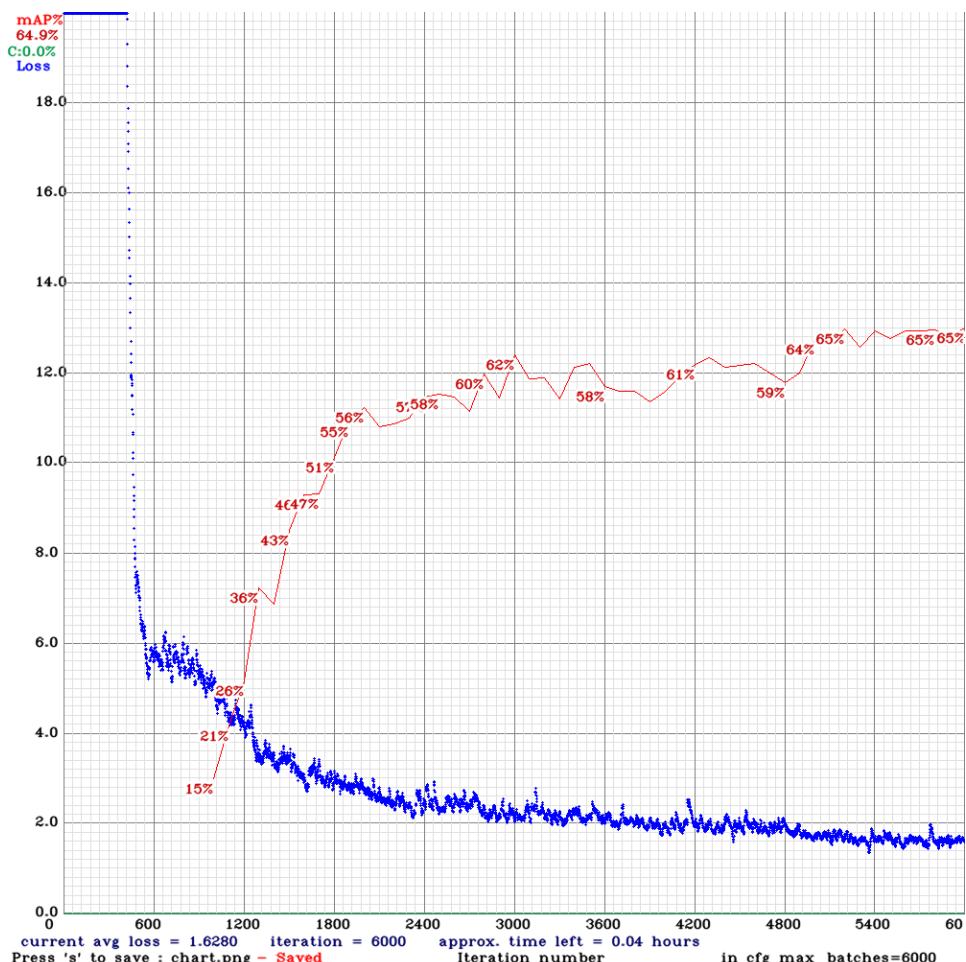
## 4.2. Eksperimen Kedua

- Hal ini dilakukan untuk mengamati pengaruh dataset yang lebih bervariasi terhadap tingkat akurasi.
- Hasil:
  - Learning rate: 0.001
  - Lama training: 2 jam 43 menit 59 detik
  - Mean AP: 65.4%
  - no\_mask ap = 45.65%
  - mask ap = 81.98%



### 4.3. Eksperimen Ketiga

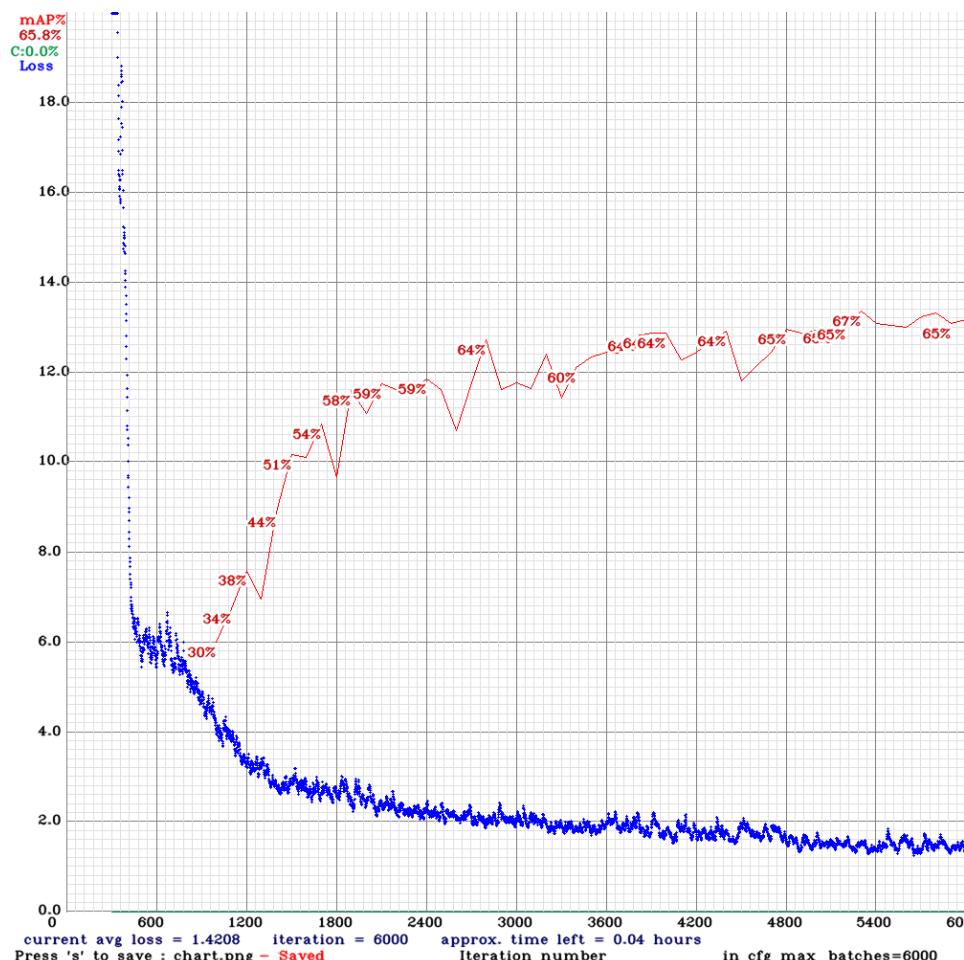
- Sama seperti eksperimen kedua (700 training images), tapi semua image diberi histogram equalization untuk mendistribusikan intensitas.
- Learning rate: 0.001
- Hasil:
  - Lama training: 2 jam 21 menit 40 detik
  - mAP: 64.9%
  - no\_mask ap = 47.36%
  - mask ap = 82.47%





#### 4.4. Eksperimen Keempat

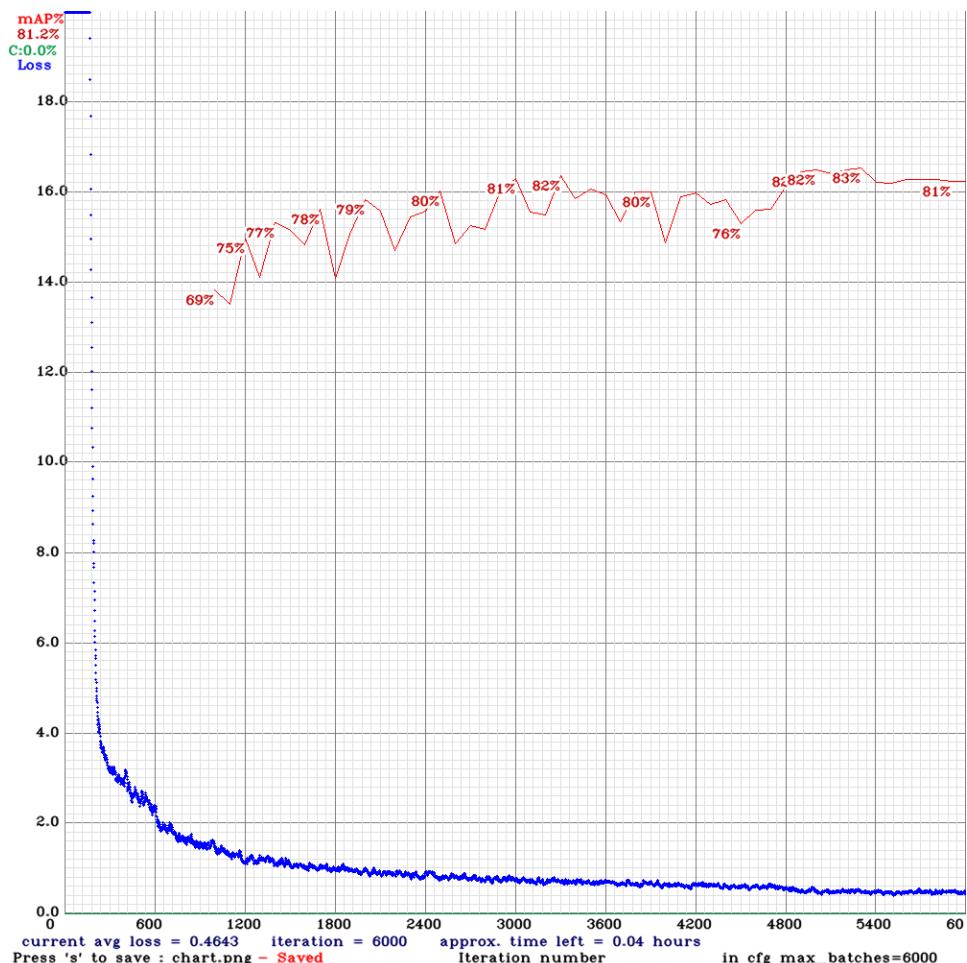
- Sama seperti eksperimen kedua, tapi learning rate digandakan (0.002)
- Peningkatan akurasi di langkah-langkah awal lebih tinggi, namun Loss rate lebih fluktuatif
- Hasil:
  - Lama training: 2 jam 34 menit
  - mAP = 65.8%
  - no\_mask ap = 46.23%
  - mask ap = 81.55%





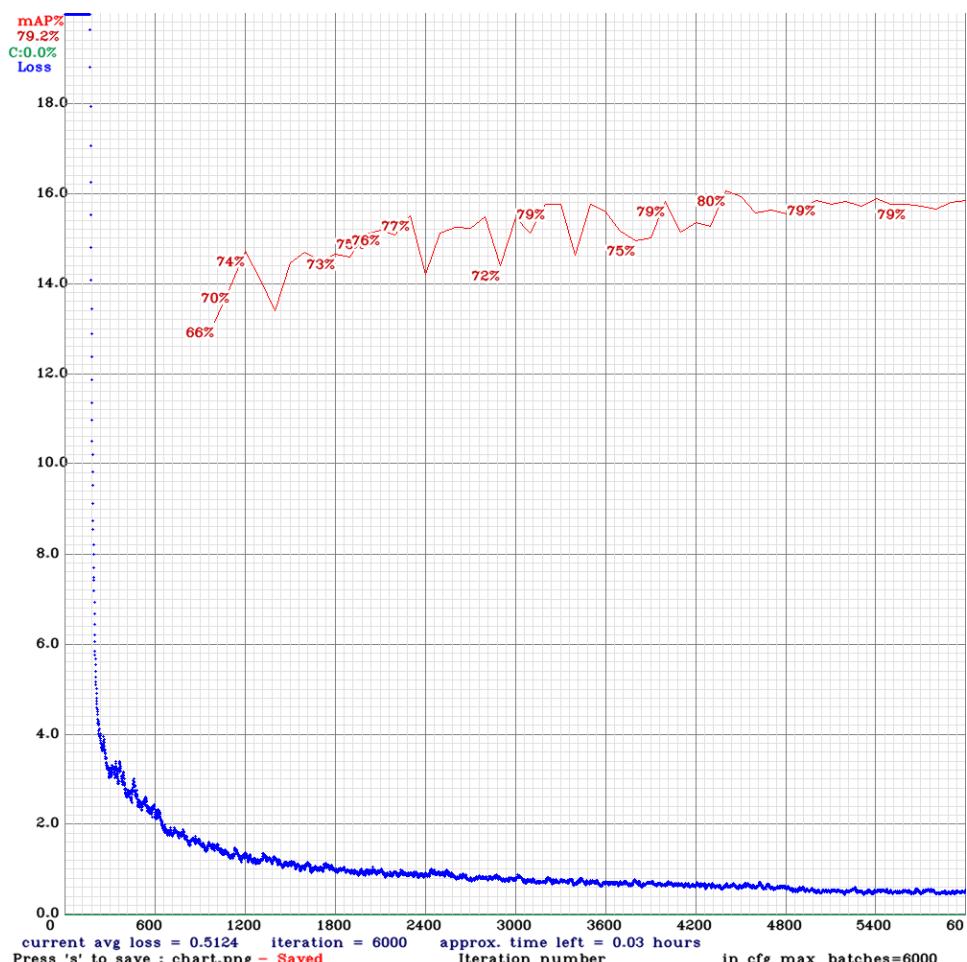
## 4.5. Eksperimen Kelima

- Sama seperti eksperimen keempat (700 *training images*, learning rate = 0.002), tapi menggunakan model Yolo V4 tiny. Hal ini dilakukan untuk mengamati tingkat akurasi Yolo v4 terhadap Yolo v3.
- Lama training: 2 jam 14 menit 4 detik
- Hasil:
  - mAP: 81.2%
  - no\_mask ap = 71.44%
  - mask ap = 89.16%



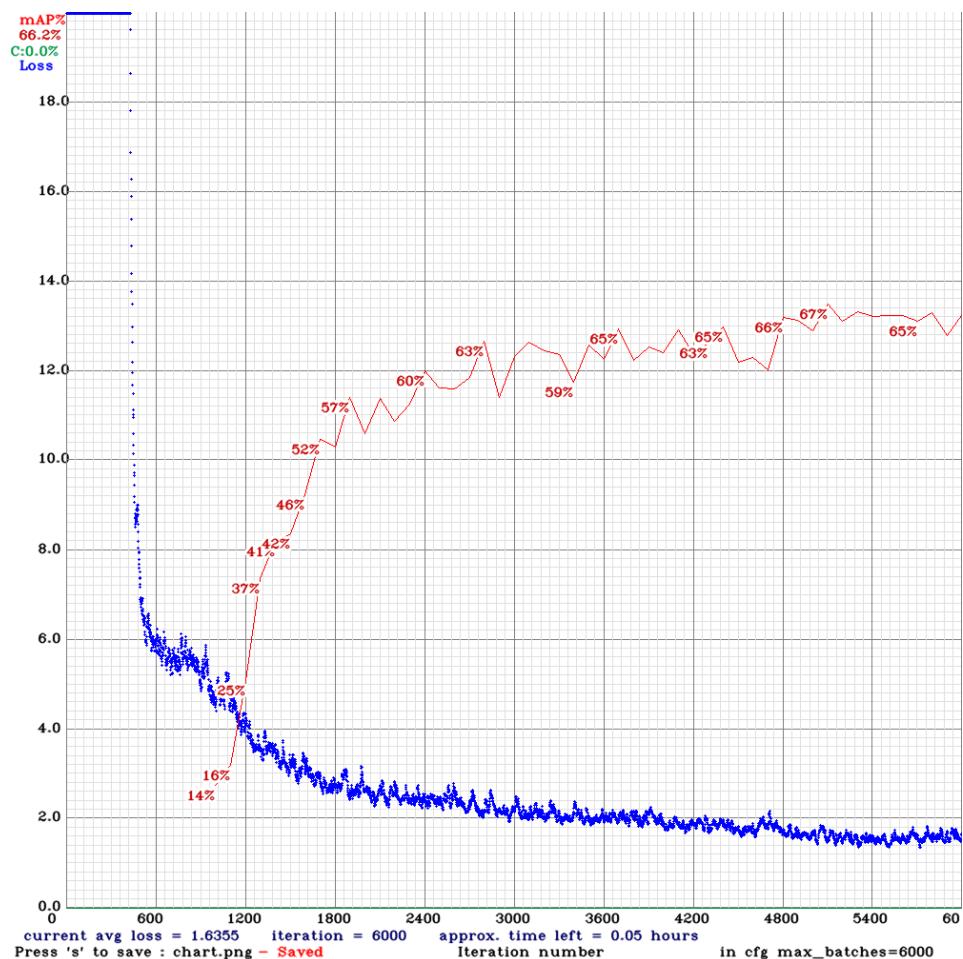
## 4.6. Eksperimen Keenam

- Sama seperti eksperimen kelima (700 training images, Yolo v4 tiny, learning rate = 0.002), tapi semua gambar input diberi *histogram equalization*
- Lama training: 1 jam 54 menit 58 detik
- Hasil
  - mAP: 79.2%
  - no\_mask ap = 68.79%
  - mask ap = 89.62%



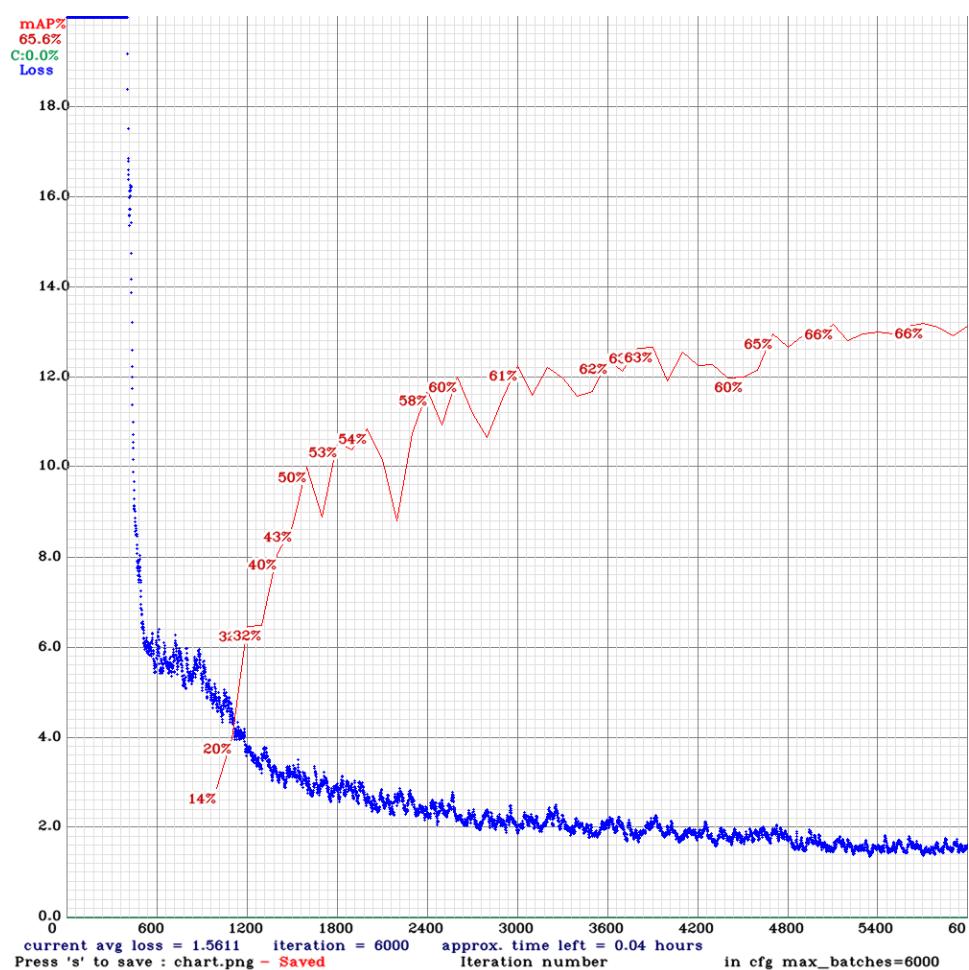
## 4.7. Eksperimen Ketujuh

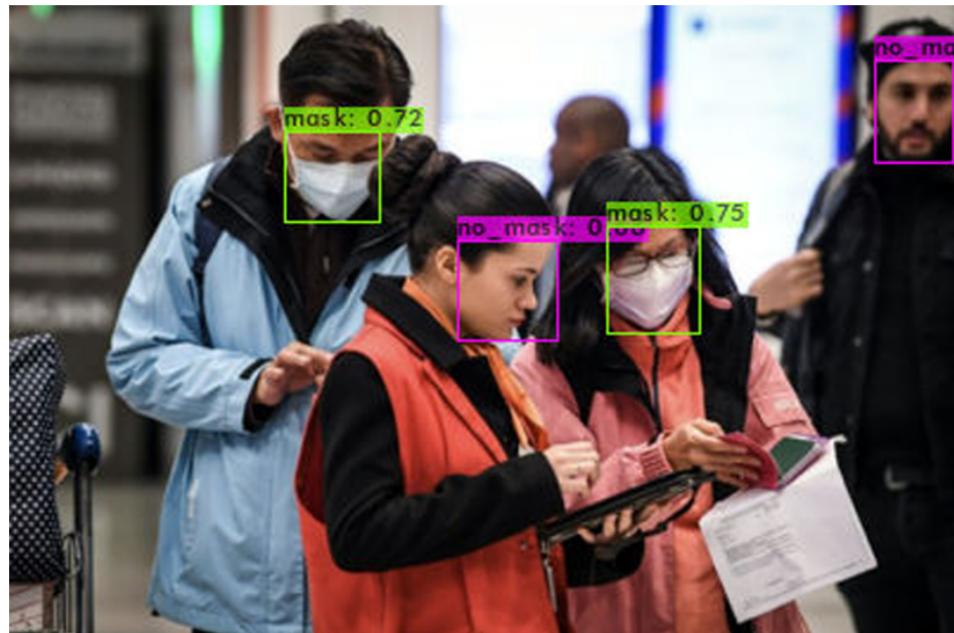
- Sama seperti eksperimen ketiga
- Learning rate: 0.001
- Lama training: 3 jam 30 menit
- mAP: 66.2%



## 4.8. Eksperimen Kedelapan

- Menggunakan obj2 fully equalized
- No random training
- Learning rate: 0.001
- Lama training: 2 jam 43 menit
- mAP: 65.6%





Mask



No Mask





# Penutup

## 5.1. Kesimpulan

Hal-hal yang diperoleh berdasarkan analisis hasil penelitian-penelitian dan *training* di atas:

- Yolo v4 memiliki tingkat akurasi lebih tinggi dan waktu *training* yang lebih cepat.
- *Preprocessing* gambar memiliki pengaruh terhadap tingkat akurasi
  - *Preprocessing* gambar *training* dan *testing* melalui *histogram equalization* menurunkan tingkat akurasi dalam dataset ini.
- *Learning rate* dapat mempengaruhi perubahan akurasi per *step*
- Dataset yang tidak seimbang dapat mempengaruhi akurasi dalam mendekripsi objek dengan kelas-kelas yang tidak dominan.
  - Akurasi deteksi wajah yang tidak menggunakan masker lebih rendah daripada akurasi deteksi wajah yang menggunakan masker. Dalam dataset yang digunakan, wajah yang tidak menggunakan masker lebih sedikit daripada wajah yang menggunakan masker.
  - Akurasi deteksi wajah yang tidak menggunakan masker pada YOLO v4 lebih tinggi daripada YOLO v3.

## 5.2. Refleksi Akademis

### 5.2.1. August

Ada beberapa hal yang diperoleh dari pelajaran ini:

- Deep Learning menggunakan *computing power* yang besar. Oleh karena itu, model yang berukuran kecil sering digunakan karena *computing cost* yang relatif rendah.
- Dataset yang tidak seimbang dan terlalu bervariasi dapat menurunkan akurasi model terutama pada deteksi kelas yang tidak dominan.

#### **5.2.2. Eric**

- Melakukan instalasi untuk Deep Learning tidak mudah meskipun mengikuti sama persis internet. Karena itu, diperlukan niat dan kesabaran agar dapat instalasi program Deep Learning di komputer.

#### **5.2.3. Kevin**

Beberapa hal yang diperoleh dalam pengerjaan proyek ini:

- Belajar untuk memahami openCV untuk dapat mengerjakan project ini
- Dataset yang tidak seimbang dapat mempengaruhi akurasi dalam mendekripsi terutama pada kelas - kelas yang tidak dominan.

#### **5.2.4. Steven**

- Walaupun memakai algoritma milik orang lain dan tidak membuat dari awal tidak semudah itu, karena memiliki banyak kendala mulai dari hardware yang mungkin tidak mumpuni untuk menjalankannya sehingga harus memakai bantuan online seperti google colab, dan walaupun begitu kita tetap harus memiliki internet yang stabil jika tidak kita tidak bisa memakai algoritma tersebut untuk melatih dan memprediksi.
- Untuk menghasilkan akurasi yang tinggi membutuhkan dataset yang seimbang dan merata dan jika kita tidak

memiliki data yang seimbang ada beberapa cara yang bisa kita lakukan untuk memanipulasi dataset untuk menghasilkan data yang lebih akurat

#### 5.2.5. Vincent

- Persebaran dataset yang digunakan dalam Deep Learning seharusnya diusahakan merata terhadap semua class. Hal ini mungkin kurang memiliki dasar teori yang kuat, namun banyak sekali pengujian dan pengalaman yang membuktikan demikian. Apabila dataset yang digunakan tidaklah merata, langkah terbaik yaitu menambah dataset yang tergolong dalam class yang underrepresented untuk menyeimbangi dataset sebelumnya.
- Dalam kasus di mana data baru sulit ditambahkan, ada beberapa langkah alternatif yang bisa diambil. Data baru bisa dimanipulasi menjadi lebih banyak dengan augmentation, atau data dari class yang overrepresented bisa dipotong. Namun, semuanya itu memiliki konsekuensinya masing-masing. Pada akhirnya, setiap langkah yang diambil tetap harus dicoba dan diuji hasilnya.
- Permasalahan ini terbatas pada lingkup Deep Learning. Metode AI lainnya bisa jadi tidak begitu mempermasalahkan persebaran dataset yang tidak merata

## Daftar Pustaka

- Bochkovskiy, A., Wang, C., & Liao, H.M. 2020. *YOLOv4: optimal speed and accuracy of object detection.* Retrieved from <https://arxiv.org/abs/2004.10934>.
- Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. 2016. *You Only Look Once: Unified, Real-Time Object Detection.* Retrieved from <https://arxiv.org/abs/1506.02640>.
- Redmon, J. & Farhadi, A. 2016. *YOLO9000: better, faster, stronger.* Retrieved from <https://arxiv.org/abs/1612.08242>.
- Redmon, J. & Farhadi, A. 2018. *YOLO v3: an incremental improvement.* Retrieved from <https://pjreddie.com/media/files/papers/YOLOv3.pdf>.

# Lampiran

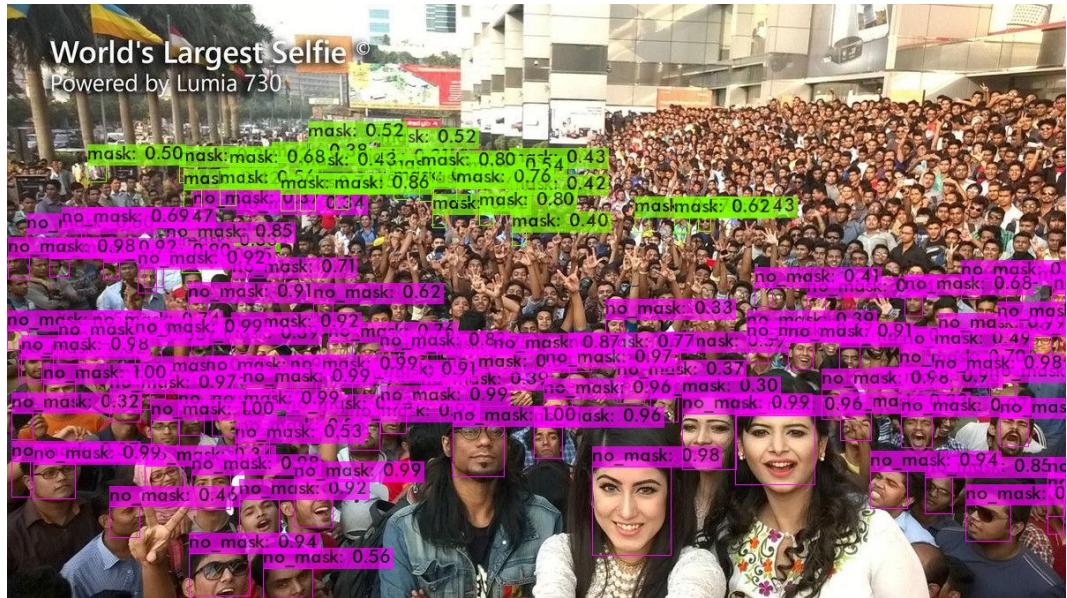
## World's Largest Selfie (YOLO v3 tiny vs YOLO v4 tiny)



Foto Asli



Hasil deteksi menggunakan model YOLO v3 tiny hasil *training* dari eksperimen kedua.



Hasil deteksi menggunakan model YOLO v4 *tiny* hasil *training* dari eksperimen kelima. YOLO v4 *tiny* dapat mendekksi lebih banyak wajah, namun terdapat banyak *false positives* (beberapa *bounding box* yang ditandai tidak mengandung bagian wajah dan semua deteksi dengan label “mask” tidak mengandung wajah yang memakai masker). Selain itu, ada beberapa *bounding box* yang tidak meliputi seluruh wajah.