IMPLEMENTASI DAN ANALISIS PERFORMANSI METODE *YOU ONLY LOOK ONCE* (YOLO) SEBAGAI SENSOR PORNOGRAFI PADA VIDEO

IMPLEMENTATION AND PERFORMANCE ANALYSIS OF YOU ONLY LOOK ONCE (YOLO) METHOD AS PORN CENSORSHIP IN VIDEO

Harits Hammam Al Asyhar[1], Suryo Adhi Wibowo[2], Gelar Budiman[3]

[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]**haritshammam@student.telkomuniversity.ac.id**, [2]**suryoadhiwibowo@telkomuniversity.co.id**,
[3]**gelarbudiman@telkomuniversity.ac.id**

Abstrak

Video yang memuat konten pornografi sangatlah mudah untuk diakses pada saat ini. Setiap orang dari berbagai macam usia dan latar belakang dapat dengan mudah untuk menontonnya. Oleh karena itu, diperlukan suatu upaya untuk mempersulit orang-orang untuk dapat menonton video yang memuat konten pornografi. Penelitian ini akan menggunakan salah satu algoritma deteksi objek, yakni *You Only Look Once* (YOLO) untuk mendeteksi adanya konten pornografi pada video untuk keperluan penyensoran. YOLO merupakan pengembangan dari algoritma deteksi objek *Convolutional Neural Network* (CNN), sehingga YOLO mampu mendeteksi suatu objek dengan tingkat akurasi yang cukup tinggi dan frame rate yang lebih tinggi dibandingkan dengan algoritma deteksi objek *state-of-the-art* lainnya. Skema *dataset* yang digunakan pada penelitian ini adalah 2507 data latih, 2507 data validasi, dan 1253 data uji. Dataset akan dibagi menjadi 3 kelas, yakni npe (*non-porn easy*), npd (*non-porn difficult*), dan *porn*. Parameter performansi yang ditinjau adalah *mean Average Precission* (mAP). Nilai mAP tertinggi sebesar 48,13% dengan konfigurasi *hyperparameter learning rate* 0,001, *epoch* 100, dan *batch size* 32 untuk proses *training*, ditambah dengan proses *fine-tuning* dengan *learning rate* awal 0,0001, *epoch* 35, dan *batch size* 4. Model tersebut berhasil diimplementasikan pada aplikasi *desktop* untuk mendeteksi sekaligus menyensor gambar pornografi pada video dengan *frame rate* rata-rata di 25 fps.

Kata kunci : ***object detection, CNN, YOLO, censorship, pornography***

Abstract

At present, video that contains pornographic content are very easy to access. People of all ages and backgrounds can easily watch it. Therefore, an effort is needed to complicate people to be able to watch pornographic videos. This research will use one of the object detection algorithms, namely You Only Look Once (YOLO) to detect the presence of pornographic content in the video for censorship purpose. YOLO is one of the developments of Convolutional Neural Network (CNN) object detection algorithm, and YOLO is able to detect an object with a fairly high accuracy and higher frame rate compared to other state-of-the-art object detectional gorithms. The model in this research will be trained by using dataset that contains 6267 images that will be classified into 3 classes, namely non-porn easy, non-porn difficult, and porn. In this research, the dataset scheme is using 2507 train data, 2507 validation data, and 1253 test data. The dataset is divided into 3 classes, namely npe (non-porn easy), npd (non-porn difficult), and porn. Performance parameter that is evaluated is mean Average Precission (mAP). The highest mAP value is 48.13% by using hyperparameter configuration of 0.001 learning rate, 100 epoch, and 32 batch size for the training phase, and with addition on fine-tuning process with 0.0001 learning rate, 35 epoch, and 4 batch size. The model is succesfully implemented on the desktop application for detecting and censoring pornographic image in a video with average frame rate of 25 fps.

Keywords: ***object detection, CNN, YOLO, censorship, pornography***

1. Pendahuluan

Proses pertukaran informasi melalui internet semakin mudah dan cepat untuk dilakukan saat ini. Namun, sejalan dengan perkembangan tersebut, konten pornografi juga semakin mudah untuk diakses oleh pengguna dari berbagai macam usia. Terdapat beragam metode[1][2] yang telah diusungkan untuk memblokir akses ke situs web yang mengandung konten pornografi. Tetapi, terdapat juga metode lain yang bisa melakukan *bypass* pada situs web yang terblokir, seperti *Virtual Private Network* (VPN)[3] yang telah lazim digunakan oleh pengguna internet saat ini. Oleh karena itu, penelitian ini mengimplementasikan salah satu metode di bidang *Computer Vision*, yakni object detection yang mampu mendeteksi sekaligus menyensor konten pornografi pada video dengan *frame rate* yang tinggi.

Computer Vision merupakan salah satu cabang dari *Artificial Intelegence* (AI), di mana sebuah komputer dilatih agar dapat melihat konten digital berupa citra ataupun video layaknya manusia, yang bertujuan untuk mengekstrak informasi yang terdapat di dalamnya[4]. Salah satu metode *Computer Vision* adalah object detection, yaitu suatu metode yang berfungsi untuk menemukan lokasi objek tertentu pada suatu citra sekaligus mengklasifikasikan objek tersebut ke dalam sebuah kelas yang telah ditentukan. Saat ini, terdapat berbagai macam metode object detection yang merupakan pengembangan dari arsitektur jaringan syaraf tiruan *Convolutional Neural Network* (CNN)[5], seperti *Faster* R-CNN[6], *Single Shot Detector* (SSD)[7], dan *You Only Look Once* (YOLO)[8]. Masing-masing metode tersebut memiliki keunggulannya masing-masing, sehingga diperlukan studi literatur mengenai penggunaan metode yang cocok pada suatu permasalahan.

Pada penelitian ini, penulis menggunakan YOLOv3[9] sebagai metode deteksi konten pornografi pada video. Alasan penulis menggunakan metode ini daripada [6] dan [7] adalah karena YOLOv3 mampu mendeteksi objek dengan *frame rate* yang lebih tinggi daripada *Faster* R-CNN[8]. Hal ini diperlukan karena pada umumnya video yang nyaman untuk ditonton memiliki nilai *frame rate* minimal sebesar 24 fps, dan juga YOLOv3 memiliki nilai parameter *Mean Average-Precision* (mAP) pada *metric Intersection over Union* (IoU) 0.5 yang lebih baik daripada SSD yang telah dilatih pada *dataset* COCO[10], yang berarti memiliki tingkat akurasi yang lebih baik dalam mengenali objek. *Hyperparameter* yang dikonfigurasi dalam penelitian ini adalah *epoch* dan *learning rate* pada masing-masing proses *training* dan *fine-tuning*. Kemudian penulis membuat 5 model *object detector* yang masing-masing memiliki konfigurasi *hyperparameter* yang berbeda untuk menguji mAP yang dihasilkan dari setiap model, dan menentukan model terbaik yang akan digunakan pada aplikasi *desktop*.

2. Tinjauan Pustaka

Pada bab ini menjelaskan mengenai definisi dan konsep dasar mengenai *object detection algorithm*, *Convolutional Neural Network* (CNN), dan *You Only Look Once* (YOLO).

2.1. *Object Detection Algorithm*

Object detection algorithm merupakan salah satu algoritma *computer vision* yang berfungsi untuk menentukan di mana lokasi objek pada suatu citra berada dengan *bounding box*, sekaligus mengklasifikasikan objek tersebut ke dalam sebuah kelas yang telah ditentukan. *Metric* yang paling cocok dan sering digunakan untuk mengukur kemampuan algoritma deteksi objek adalah *Mean Average-Precission* (mAP). Terdapat berbagai macam jenis arsitektur *object detection algorithm*, seiring waktu, CNN telah menjadi arsitektur dasar yang paling sering digunakan karena keunggulanannya dalam melakukan deteksi objek pada suatu citra[11].

2.2. *Convolutional Neural Network* (CNN)

CNN merupakan arsitektur dasar yang digunakan pada bidang *deep learning* yang dapat digunakan untuk melakukan klasifikasi, *image segmentation*, dan *object detection* pada citra[11]. Pada CNN, nilai pixel citra yang direpresentasikan dalam matriks berdimensi sama dengan resolusi citra, dikonvolusikan dengan matriks filter berukuran *fxf*. Nilai dari filter inilah yang akan menentukan hasil ekstraksi fitur setelah proses konvolusi.

2.2.1. *Padding*

Padding merupakan sekumpulan nilai 0, yang berperan seperti *border* pada matriks masukan yang berfungsi agar dimensi matriks hasil konvolusi sama dengan dimensi matriks masukan, sehingga tidak terjadi kehilangan informasi pada matriks hasil konvolusi. Nilai *padding* menentukan berapa banyak 'lapis' *border* tersebut.

2.2.2. *Stride*

Stride merupakan jarak pergeseran filter terhadap matriks citra masukan ketika melakukan operasi konvolusi. Arah pergeseran filter sebesar *s* kolom ke kanan hingga akhir kolom dan berlanjut ke *s* baris di bawahnya. *Stride* dapat berguna untuk menentukan dimensi *output matrix*.

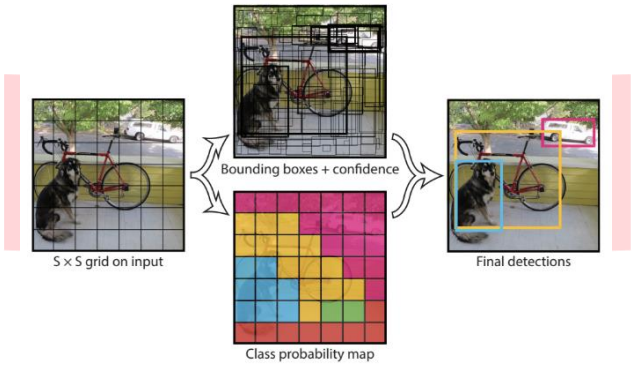
2.2.3. *Backpropagation*

Seluruh proses yang dimulai dari *convolution layer* hingga *fully connected layer* secara berurutan dinamakan proses *forward pass*. Pada permulaan proses training, nilai *weight* diinisialisasikan secara acak, dan akan menghasilkan hasil klasifikasi yang kurang optimal. Untuk itu, perlu dilakukan *backpropagation*, yakni proses pembaruan nilai *weight*, yang dimulai dari *fully connected layer* hingga ke awal *convolutional layer* pada sebuah CNN. *Backpropagation* sendiri merupakan sebuah algoritma yang menghitung *error gradients* dari *weight*, kemudian nilai *gradients* tersebut akan digunakan pada algoritma optimasi bernama *Gradient Descent*, yang akan melakukan pembaruan *weight* secara bersamaan, sehingga diharapkan mendapat hasil klasifikasi yang lebih optimal dari yang sebelumnya.

2.3. *You Only Look Once (YOLO)*

YOLO merupakan salah satu *state-of-the-art object detection algorithm* yang membagi citra masukan ke dalam suatu *grid* berukuran *SxS*. Ukuran dari *grid cell* tersebut tergantung pada *input size* yang digunakan pada suatu arsitektur. Pada YOLOv3, jika *input size* 416x416, maka ukuran *grid size* adalah 13x13, 26x26, dan 52x52. Setiap sel bertugas untuk memprediksi objek yang ada di dalam sel tersebut dengan *bounding box* beserta *confidence* yang merupakan nilai probabilitas keberadaan suatu objek pada bounding box tersebut. Kemudian, setelah *bounding box* dipetakan berdasarkan nilai *confidence* yang dihasilkan, YOLO akan memprediksi kelas dari objek yang terdapat pada *bounding box* tersebut beserta probabilitasnya, sehingga terbentuklah class *probability map*.

Dari sekian banyak *bounding box* yang dihasilkan, untuk mendapatkan *bounding box* beserta kelas objek dengan probabilitas yang tinggi, maka dari seluruh hasil prediksi tersebut, hanya yang melampaui *threshold* saja yang akan digunakan. Jika terdapat duplikasi *bounding box*, maka *Non-max Suppresion* (NMS) berperan untuk menghilangkan duplikat tersebut. Gambar 1 merupakan ilustrasi dari cara kerja YOLO.



Gambar 1. Ilustrasi algoritma YOLO[12].

Pada YOLOv3, *feature extractor* yang sebelumnya menggunakan Darknet-19 menjadi Darknet-53, dan juga proses deteksi objek yang kini menggunakan 3 skala di mana pada setiap skala menggunakan 3 *anchor boxes* sehingga berdampak pada meningkatnya kemampuan mendeteksi objek yang lebih kecil[13].

2.4. NPDI *Pornography Dataset*

Dataset yang digunakan dalam penelitian ini adalah [14] yang telah disusun oleh *University* of Campinas, Brazil. Dataset ini terdiri 16.727 frame dari 800 video, yang kemudian terbagi menjadi 3 class, yakni *non-porn easy*, *non-porn difficult*, dan *porn* sebanyak 6785 citra, 3555 citra, dan 6387 citra secara berturut-turut.

2.5. *Epoch*

Untuk mendapatkan hasil klasifikasi yang lebih optimal, *gradient descent* akan melakukan pembaruan nilai *weight* dari sebuah *neural network* pada saat proses *backpropagation*. Serangkaian proses *forward pass* hingga *backpropagation* terhitung sebagai 1 *epoch*. Untuk mendapatkan nilai *weigh* tyang lebih tepat lagi, tentunya dibutuhkan lebih dari satu kali proses pembaruan, atau lebih dari 1 *epoch*.

2.6. *Batch Size*

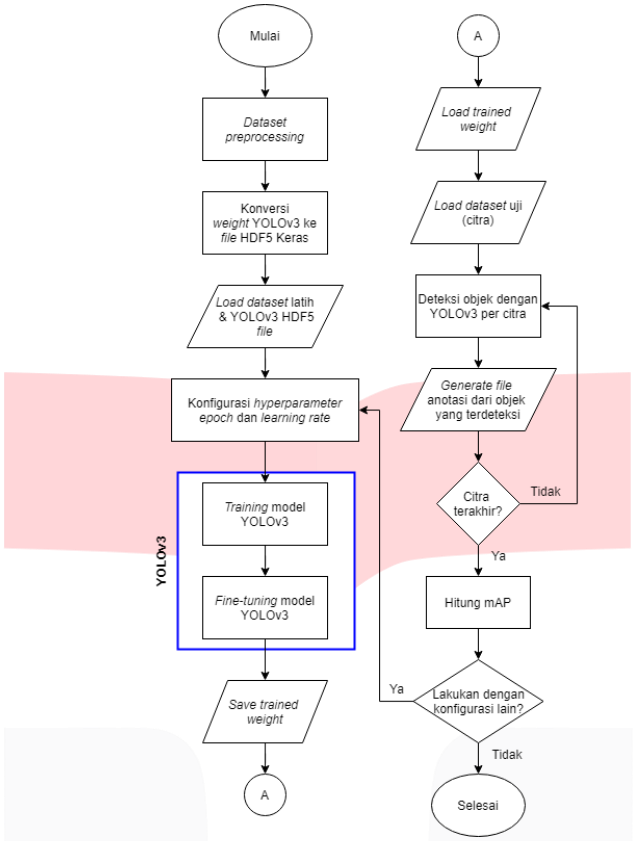
Pada saat proses *training* model, data latih yang sangat banyak dapat dibagi menjadi beberapa bagian dalam satu kali *epoch* untuk mengurangi kebutuhan komputasi yang terlalu besar, yang dinamakan dengan *batch size*. Maka, semakin besar *batch size* yang digunakan, maka kebutuhan komputasi pada *hardware* juga akan semakin besar.

2.7. *Learning Rate*

Hyperparameter learning rate memiliki peranan sebagai penentu seberapa banyak porsi *weight* yang akan diperbarui ketika proses *backpropagation*. *Learning rate* memiliki rentang nilai di antara 0 hingga 1. Ketika nilai *learning rate* yang diberikan terlalu besar, maka proses *training* akan lebih cepat, namun akan membuat *training error* senantiasa meningkat, sehingga *final weight* akan menjadi kurang optimal. Berbeda halnya dengan nilai *learning rate* yang terlalu kecil, tidak hanya proses *training* akan menjadi sangat lama, tetapi juga akan menghasilkan *training error* yang tinggi secara permanen, sehingga akan menghasilkan *final weight* yang kurang optimal pula[14].

3. Desain Model Sistem

Pada penilitian ini, proses pelatihan hingga pengujian model akan berjalan pada Ananconda *virtual environment* dengan menggunakan bahasa pemrograman Python versi 3.7.4 dan TensorFlow 1.14 di dalamnya. Skema proses pelatihan hingga pengujian model dapat dilihat pada gambar 3.



Gambar 2. Flowchart proses *training* dan *testing* model.

3.1. *Dataset Preprocessing*

Langkah pertama yang penulis lakukan pada tahapan ini adalah dengan mengumpulkan *dataset* berupa citra yang akan digunakan sebagai data latih dan data uji. Skema pembagian *dataset* ditunjukkan pada Tabel 1.

Tabel 1. Skema *Dataset*

Kelas	Data Latih	Data Validasi	Data Uji
npe	690	690	417
npd	745	745	418
porn	1072	1072	418
Jumlah	5014		1253
Split Ratio	80%		20%
	50%	50%	

3.2. Konversi *Pre-trained Weight* YOLOv3

File *pre-trained weight* YOLOv3 yang telah diunduh kemudian dikonversi ke dalam *file* Keras HDF5. Pengkonversian *format file* ini dilakukan karena *file weight* yang didukung pada arsitektur YOLOv3 pada aplikasi pelatihan model TensorFlow 1.14 ini adalah HDF5.

3.3. *Load Dataset* Latih dan *Weight* HDF5 YOLOv3

Dataset latih beserta label *ground truth* yang sudah dibuat sebelumnya kemudian dikumpulkan menjadi satu *file* anotasi berformat .txt yang setiap barisnya menunjukkan lokasi masing-masing citra latih, lokasi 4 titik (x-*min*, x-*max*, y-*min*, dan y-*max*) *ground truth* pada piksel yang dipetakan pada koordinat sebesar resolusi citra, dan juga kelas objek yang telah ditetapkan sebelumnya. Selain *dataset* latih, *file weight* HDF5 YOLOv3 juga akan dimuat pada program pelatihan model.

3.4. Skema Konfigurasi *Hyperparameter* Model

Pada arsitektur yang penulis gunakan, terdapat 3 *hyperparameter* yang dapat dikostumisasi, yakni *epoch*, *learning rate*, dan *batch size*. Untuk ukuran *batch size* yang digunakan, penulis menggunakan *batch size* 32 untuk proses *training* dan 4 untuk proses *fine-tuning* pada kelima model. Sedangkan untuk nilai dari *step size* pada training dan *fine-tuning* didapatkan dengan membagi jumlah data latih (tanpa data validasi) dengan *batch size*. Tabel 2 merupakan skema pembagian konfigurasi model yang penulis rancang.

Tabel 2. Skema Konfigurasi *Hyperparameter*

Skema	*Training Epoch*	*Training Learning Rate*	*Fine-tuning Epoch*	*Fine-tuning Learning Rate*
A	50	0,001	50	0,0001
B	50	0,0001	50	0,00001
C	100	0,001	100	0,0001
D	100	0,0001	100	0,00001
E	200	0,001	200	0,0001

3.5. Melatih Model

Dalam mendeteksi objek, pada Darknet-53 terdapat 2 proses yang terjadi, yakni ekstraksi fitur dan *detector*. Citra masukan akan diekstrak fiturnya menggunakan Darknet-53 pada 75 *layer* pertama, kemudian setelah fitur tersebut didapatkan, akan dijadikan sebagai input ke *detector* yang berfungsi untuk memprediksi *bounding box* dan *class* objek yang terdeteksi berdasarkan fitur tersebut. Selama proses pelatihan model ini, nilai *weight* dari *file weight* HDF5 YOLOv3 yang telah di*load* tadi akan secara terus-menerus di*update* sebanyak nilai *training epoch* yang telah diberikan saat konfigurasi *hyperparameter* menggunakan algoritma *backpropagation*.

Setelah proses pelatihan selesai, akan dilakukan proses *fine-tuning* yang berfungsi untuk mengoptimasi model yang telah dilatih dengan meminimalisi kemungkinan *overfitting* sekecil mungkin. Pada proses ini juga, *weight* dari proses pelatihan secara terus-menerus diupdate sebanyak nilai *fine-tuning epoch*. Oleh karena itu, jika model akan mengalami *overfitting*, maka proses *fine-tuning* akan secara otomatis berhenti, walaupun proses pembaruan *weight* belum berulang sampai sebanyak nilai *fine-tuning epoch* yang diberikan di awal.

3.6. Pengujian Model

Setelah proses *training* model deteksi objek selesai, model yang berupa *file weight* HDF5 tersebut akan digunakan untuk mendeteksi objek yang termasuk ke dalam kategori konten pornografi pada dataset uji yang berupa citra. Hasil dari deteksi citra tersebut kemudian akan digunakan untuk menghitung nilai mAP.

3.7. Pengimplementasian Model

Setelah perhitungan nilai mAP kelima model yang telah dilatih, penulis kemudian menentukan model yang memiliki *mean Average Precission* (mAP). Aplikasi berbasis desktop ini penulis rancang menggunakan *library* Python bernama TkInter yang memiliki kegunaan khusus untuk membangun aplikasi GUI berbasis *desktop*. Aplikasi ini bertujuan untuk memudahkan pengguna ketika akan melakukan penyensoran konten pornografi pada suatu video tanpa harus mengetikkan perintah pada command prompt, dengan cara cukup memilih video dan direktori tujuan penyimpanan video yang tersensor secara langsung dengan cukup menekan tombol. Namun, aplikasi ini hanya bertindak sebagai *shortcut*, di mana proses pendeteksian objek tetap berjalan menggunakan TensorFlow pada *command prompt*.

4. Analisis Parameter Kinerja

Mean average precision merupakan parameter utama yang digunakan sebagai tolak ukur akurasi model yang telah dilatih menggunakan *dataset* tertentu. *Mean average precission* merata-ratakan nilai AP dari setiap kelas yang ada pada model yang telah diuji. Untuk mengetahui nilai AP suatu kelas, perlu diketahui nilai *precission* dan *recall* yang dapat dihitung menggunakan persamaan:

$$Precission = \frac{TP}{TP+FP}, \; Recall = \frac{TP}{TP+FN} \tag{1}$$

Setelah itu, nilai *precission* di*plot* terhadap nilai *recall* untuk mendapatkan pola yang berbentuk zig-zag. Untuk menghitung nilai AP, hasil *plot* berbentuk zig-zag tersebut akan dihaluskan, di mana nilai *precission* (*p*) pada suatu *recall* (*r*) akan disamakan dengan nilai *precission* maksimum pada *recall* selanjutnya (*r*'), atau dituliskan menggunakan persamaan berikut:

$$p_{inter}(r_{n+1}) = \max p(r'); \; r' \geq r_{n+1}. \tag{2}$$

Pada PASCAL VOC2012, nilai AP didapatkan dengan menghitung area kurva pada nilai *recalls* yang unik, yakni ketika nilai *precission* maksimum jatuh. Sehingga area yang terhitung merupakan area di mana zig-zag dihaluskan, yang disebut dengan *Area Under Curve* (AUC). Selanjutnya, nilai AP didapatkan menggunakan persamaan berikut:

$$AP = \sum(r_{n+1} - r_n)\, p_{inter}(r_{n+1}). \quad (3)$$

Kemudian untuk mendapatkan nilai mAP dapat dihitung menggunakan persamaan:

$$mAP = \sum_{i=1}^{N} \frac{AP(i)}{N} \times 100\%, \quad (4)$$

di mana *N* merupakan banyaknya kelas yang terdapat pada suatu model yang telah dilatih.

5. Hasil dan Analisis

Pada bab ini, penulis akan melakukan pengujian model dan analisis hasil pengujian menggunakan parameter performansi *Mean Average Precission* (mAP), yang merupakan nilai rata-rata dari *Average Precission* setiap kelas pada suatu model yang telah dilatih.

5.1. Skenario Pengujian Sistem

Setelah proses pelatihan model selesai, didapatkan 5 model dalam bentuk *file weight* HDF5 dengan 5 skema konfigurasi *hyperparameter* yang berbeda-beda. Pada proses pelatihan di setiap model, terdapat 2 buah *file weight* yang dihasilkan, yakni *weight* dari proses *training*, dan *final weight* dari proses *fine-tuning* yang merupakan proses penyempurnaan dari *weight* saat *training*, sehingga total *weight* keseluruhan adalah sebanyak 10 *file*. Setelah itu, pengujian dilakukan dengan cara menghitung mAP dari 10 *weight* tersebut, dan didapatkan nilai AP dari masing-masing kelas pada setiap model.

6. Data Hasil Pengujian Sistem

Berikut merupakan tabel perbandingan nilai mAP dari seluruh model saat proses *traning* yang menggunakan *batch size* 32 dan *step size* 78, dan saat proses *fine-tuning* yang menggunakan *batch size* 4 dan *step size* 626 di kelima model. Hasil pengujian mAP dari proses *training* dapat dilihat pada tabel 3. dan untuk *fine-tuning* dapat dilihat pada tabel 4.

Tabel 3. Hasil Pengujian mAP setelah *training*

Skema	mAP (%)	*Average Precission*		
		porn	**npe**	***npd***
A	**39,15**	0,55	0,14	0,48
B	30,70	0,51	0,14	0,27
C	36,66	0,50	0,14	0,46
D	35,41	0,54	0,13	0,39
E	36,29	0,52	0,16	0,41

Berdasarkan tabel diatas, dapat dilihat bahwa dari hasil *training*, nilai mAP tertinggi yang dihasilkan hanya sebesar 39,15% pada model yang menggunakan skema *A*, sedangkan nilai mAP terendah yang dihasilkan sebesar 30,70% pada model yang menggunakan skema *B*. Kemudian setelah proses *fine-tuning*, terdapat peningkatan nilai mAP sebesar rata-rata 10,74% sehingga kelima model dapat memiliki mAP di atas 40%. Berikut merupakan tabel dari nilai mAP setelah *fine-tuning*.

Tabel 4. Hasil Pengujian mAP setelah *fine-tuning*

Skema	mAP (%)	*Average Precission*		
		porn	**npe**	***npd***
A	47,42	0,50	0,16	0,76
B	44,99	0,50	0,16	0,70
C	**48,13**	0,51	0,18	0,75
D	44,11	0,51	0,15	0,67
E	47,26	0,50	0,16	0,76

Berdasarkan tabel di atas, nilai mAP tertinggi dimiliki oleh model yang menggunakan skema *C* setelah dilakukan proses *fine-tuning*, yakni sebesar 48,13%. Sedangkan untuk model final yang memiliki mAP terendah

adalah model *D*, yakni sebesar 44,11%. Dapat diamati pada tabel 4 bahwa pada kelas npd mengalami kenaikan nilai AP yang cukup drastis sebesar 32,6% rata-rata kenaikan, sehingga hal tersebutlah yang mempengaruhi meningkatnya nilai mAP secara keseluruhan pada kelima model setelah proses *fine-tuning*.

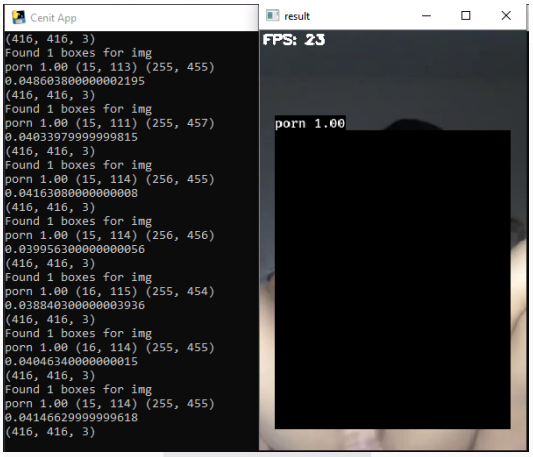
Perlu diketahui bahwasanya saat proses *fine-tuning*, terjadi penurunan nilai *learning rate* secara otomatis oleh TensorFlow untuk mendapatkan nilai *training loss* dan *validation loss* yang lebih kecil dari *epoch* yang sebelumnya. Ketika kedua nilai tersebut tidak dapat diperkecil lagi bahkan dengan nilai *learning rate* yang lebih kecil dari yang sebelumnya, maka TensorFlow akan menghentikan proses *fine-tuning* meskipun belum mencapai *epoch* yang telah diberikan guna menghindari terjadinya *overfitting*, sehingga didapatkan model dengan *weight* yang paling optimal. Sehingga pada system ini, konfigurasi pada *hyperparameter learning rate*lah yang mempengaruhi nilai mAP yang akan didapatkan pada suatu model. Berikut tabel yang menunjukkan *epoch* dan penyesuaian *learning rate* saat proses *fine-tuning*.

Tabel 5. Penyesuaian nilai *hyperparameter* saat *fine-tuning*

Skema	***Initial Epoch***	***Used Epoch***	***Initial Learning Rate***	***Last Learning Rate***
A	50	21	1*e*-4	1*e*-10
B	50	31	1*e*-5	1*e*-12
C	100	35	1*e*-4	1*e*-11
D	100	16	1*e*-5	1*e*-9
E	200	15	1*e*-4	1*e*-7

6.1 Pengimplementasian Model

Berdasarkan hasil pengujian nilai mAP, model yang dilatih dari skema C memiliki nilai mAP terbaik diantara keempat skema lainnya. Oleh karena itu, model C inilah yang akan penulis gunakan sebagai *object detector* untuk diimplementasikan pada aplikasi *desktop* yang telah penulis rancang yang bernama Cenit.



Gambar 3. Tampilan aplikasi Cenit saat mendeteksi konten pornografi.

Gambar 3 merupakan tampilan aplikasi Cenit saat *object detector* dijalankan. Terdapat 2 *window* yang terdapat pada layar *desktop*, yang pertama merupakan *window command prompt* di mana TensorFlow berjalan, menampilkan beberapa informasi berupa *class*, *confidence*, lokasi objek yang terdeteksi oleh model yang ditandai dengan koordinat *bounding box*, lamanya waktu objek terdeteksi dalam satuan *millisecond*, dan *input size* dari video berupa citra *frame* sebesar 416x416x3. Kemudian pada *window* yang kedua, merupakan *preview* dari video yang sedang dideteksi oleh *object detector*. Pada gambar 5, aplikasi mendeteksi objek pornografi dan melakukan penyensoran dengan *bounding box* berwarna hitam beserta nama kelas yang terdeteksi dan *confidence value* sebesar 1.00 atau 100%. *Window* ini akan terus terbuka dan berjalan selama durasi video masukan, tetapi dapat diakhiri sesuai keinginan user dengan menekan tombol ‘Q’ pada *keyboard*, dan video yang tersensor akan tersimpan pada direktori yang telah dipilih sebelumnya.

7. Kesimpulan

Arsitektur YOLOv3 mampu melakukan deteksi sekaligus menyensor objek yang termasuk ke dalam konten pornografi dengan nama kelas porn, dengan hasil pengujian terbaik menggunakan parameter *mean average precission* (mAP) sebesar 48,13%. Model yang telah dilatih mampu mendeteksi video masukan dengan *frame rate* sebesar 25 fps. Konfigurasi *hyperparameter* yang digunakan adalah *training* & *fine-tuning batch size* 32 & 4 berturut-turut, *training* & *fine-tuning epoch* sebanyak 100, *training learning rate* 0,001, dan *fine-tuning learning*

rate 0,0001. Selama proses *fine-tuning* berlangsung dengan menggunakan konfigurasi tersebut, terjadi penyesuaian *learning rate* sebanyak 7 kali penurunan pada *epoch* tertentu, di mana banyaknya nilai turun adalah sebesar 0,1 kali dari *learning rate* yang digunakan sebelumnya, hingga proses *fine-tuning* berakhir pada *epoch* ke-35.

Berdasarkan hasil pengujian, setelah proses *fine-tuning* dilakukan dengan konfigurasi *hyperparameter* apapun, nilai *average precission* (AP) tertinggi diraih oleh kelas npd dengan rata-rata 72,8%, kemudian kelas *porn* dengan rata-rata 50,4%, dan paling rendah dimiliki oleh kelas npe dengan rata-rata hanya sebesar 16,2%. Tidak terjadi peningkatan atau penurunan nilai AP yang signifikan pada kelas *porn* dan npe setelah proses *training* dan *fine-tuning*. Tetapi pada kelas npd, terjadi peningkatan nilai AP yang signifikan, yakni rata-rata sebesar 32,6%. Hal tersebutlah yang mempengaruhi peningkatan nilai mAP setelah dilakukan *fine-tuning* model, karena nilai mAP merupakan rata-rata nilai AP setiap kelas pada suatu model yang diuji.

Konfigurasi *fine-tuning epoch* hanya untuk menetapkan nilai batas maksimum *epoch* yang akan digunakan, karena pada sistem yang dirancang ini, model akan melakukan *early stopping* ketika nilai *validation loss* tidak dapat turun lagi setelah 10 kali *epoch*, sehingga walaupun *fine-tuning epoch* belum tercapai, sistem akan menghentikan proses *fine-tuning* untuk menghindari terjadinya *overfitting* pada model. Pada penelitian ini, hyperparameter yang mempengaruhi nilai mAP yang dihasilkan adalah *learning rate*.

Daftar Pustaka:

[1] H. Yu, R. Cong, L. Chen and Z. Lei, ”Blocking pornographic, illegal websites by internet host domain using FPGA and Bloom Filter,” 2010 2nd IEEE International Conference on Network Infrastructure and Digital Content, Beijing, 2010, pp. 619-623.

[2] T. Dinh, T. Ngo and D. Vu, ”A Model for Automatically Detecting and Blocking Pornographic Websites, ”2015 Seventh International Conference on Knowledge and Systems Engineering (KSE), Ho Chi Minh City, 2015, pp.244-249.

[3] Bhattarai, Saugat & Nepal, Sushil. (2016). VPN research (Term Paper).10.13140/RG.2.1.4215.8160.

[4] Simon J.D. Prince, Computer Vision: Models, Learning, and Inference, 1st ed. Cambridge, England: Cambridge University Press, 2012, pp. 16.

[5] S. Mane and P. S. Mangale, ”Moving Object Detection and Tracking Using Convolutional Neural Networks, ”2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2018, pp. 1809-1813.

[6] B. Liu, W. Zhao and Q. Sun, ”Study of object detection based on FasterR-CNN, ” 2017 Chinese Automation Congress (CAC), Jinan, 2017, pp.6233-6236.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg.SSD: Single Shot MultiBox Detector. arXiv e-prints arXiv:1512.02325v5, 2016.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. arXiv e-prints arXiv:1506.02640v5, 2016.

[9] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. arXive-prints arXiv:1804.02767, 2018.

[10] T.-Y Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, P. Dollar. Microsoft COCO: Common Objects inContext.arXiv e-prints arXiv:1405.0312, 2014.

[11] S. Hayat, S. Kun, Z. Tengtao, Y. Yu, T. Tu and Y. Du, ”A Deep Learning Framework Using Convolutional Neural Network for Multi-Class Object Recognition, ”2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, 2018, pp. 194-198.

[12] Working of YOLO (2019) [Online]. Tersedia di https://towardsdatascience.com/computer-vision-a-journey-from-cnn-to-mask-r-cnn-and-yolo-part-2-b0b9e67762b1, Diakses pada 28 November 2019.

[13] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. arXive-prints arXiv:1804.02767, 2018.

[14] Sandra Avila, Nicolas Thome, Matthieu Cord, Eduardo Valle, Arnaldo de A. Araujo. Pooling in Image Representation: The Visual Codeword Point of View. Computer Vision and Image Understanding (CVIU), volume 117, issue 5, p.453-465, 2013.

[15] Ian Goodfellow, Yoshua Bengio, Aaron CourvilleDeep Learning, Cambridge: MIT Press, 2016, pp. 429.