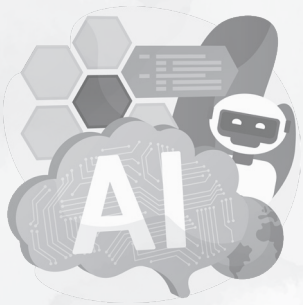


CHAPTER 2

INDUSTRIAL AI: TEXTILE DEFECT DETECTION SYSTEM

Muhammad Aldo Aditiya Nugroho, Jeffry Khosasi, Christoporus Deo Putratama, Saqfi Ahmad Rabbani, Dariel Valdano, & Nanang Cahyadi

Riset AI/PT Riset Kecerdasan Buatan

ABSTRAK

Textile Defect Detection System adalah sistem yang dipasang pada mesin inspeksi—mesin dengan kain yang terus berjalan dan biasanya akan ada operator yang memeriksa cacat pada kain tersebut—guna mendeteksi jenis cacat secara otomatis. Catatan hasil cacat yang terdeteksi dapat digunakan untuk beberapa hal, seperti penentuan *grade* dari kain atau penentuan masalah pada mesin penenun kain. Sistem ini dikembangkan dengan teknologi berbasis *computer vision*. Proses yang harus dilalui untuk membuat sistem ini adalah instalasi lingkungan penangkap gambar, seperti lampu, kamera, dan *encoder*. Selanjutnya, dilakukan pengumpulan dataset kain dengan semua jenis cacat dan anotasi dataset sembari melakukan eksplorasi model yang akan digunakan. Setelah itu, dilakukan *training* model sampai model mendapat akurasi yang diinginkan. Terakhir, model tersebut di-*deploy* pada server. *Textile Defect Detection System* cukup efektif dalam membantu mempercepat proses inspeksi karena tidak dibutuhkan waktu istirahat dan tidak memerlukan waktu libur. Terlebih, akurasi dari sistem ini dapat dikembangkan, bahkan lebih dari inspeksi yang dilakukan oleh manusia.

Kata kunci: deteksi cacat kain, *deep learning*, *computer vision*, industri tekstil

A. PENDAHULUAN

Industri tekstil merupakan salah satu dari 5 industri yang difokuskan oleh Kemenperin dalam peta jalan (*roadmap) Making Indonesia 4.0*. Hal ini karena industri tekstil menjadi salah satu industri yang berkontribusi terbesar terhadap kenaikan PDB 2017–2018 [1]. Pandemi Covid-19 menyebabkan kontraksi industri tekstil pada kuartal I/2021 sebesar -13,28% [2]. Untuk turut membantu pemulihan pasca-Covid, Riset AI percaya bahwa dibutuhkan peningkatan efisiensi agar industri tekstil dapat kembali menjadi titik kuat perindustrian Indonesia. Kecerdasan artifisial diharapkan dapat menjadi *enabler* untuk membantu industri tekstil meningkatkan

M. A. A. Nugroho, J. Khosasi, Ch. D. Putratama, S. A. Rabbani, D. Valdano,& N. Cahyadi
PT Riset Kecerdasan Buatan, e-mail: aldoan100@gmail.com

M. A. A. Nugroho, J. Khosasi, Ch. D. Putratama, S. A. Rabbani, D. Valdano, and N. Cahyadi, "Industrial AI: Textile defect detection system," in *Prosiding Use Cases Artificial Intelligence Indonesia: Embracing Collaboration for Research and Industrial Innovation in Artificial Intelligence*, B. R. Trilaksono, H. Riza, A. Jarin, N. D. S. Darmayanti, and S. Liawatimena, Eds. Jakarta: Penerbit BRIN, Februari 2023, ch. 2, pp. 15-38, doi: 10.55981/brin.668.c534
ISBN: 978-623-8052-49-3, E-ISBN: 978-623-8052-50-9

mutu dalam bersaing pada pasar global. Peningkatan kualitas sumber daya manusia dapat dimaksimalkan dengan mengalihkan pekerjaan yang bersifat repetitif pada teknologi kecerdasan artifisial.

Beberapa tantangan yang harus dihadapi untuk mengembangkan produk kecerdasan artifisial dalam konteks kolaboratif industri adalah sebagai berikut.

1. Diperlukan keahlian (*expertise)* bisnis sesuai konteks industri terkait agar implementasi solusi AI tepat guna dalam menyelesaikan masalah yang ada;
2. Diperlukan data dalam jumlah banyak;
3. Diperlukan lingkungan untuk melakukan iterasi riset;

Oleh karena itu, Riset AI bermitra dengan pemain dalam industri tekstil, yakni Sinaran Denim. Perusahaan produksi kain denim yang sudah berdiri sejak tahun 1992 ini memiliki pabrik di daerah Majalaya, Bandung, dan sudah mempekerjakan ratusan operator untuk mengoperasikan pabrik dalam keseharian.

B. STUDI LITERATUR

Berikut ini studi literatur terkait produk pendeteksi cacat pada tekstil.

1. Advantech

Pengembangan produk menggunakan model kecerdasan buatan dan diintegrasi dengan lengan robot untuk memperbaiki cacat pada kain saat cacat ditemukan. Perbedaan material kain akan diproses menggunakan model kecerdasan buatan yang berbeda [3].

2. BMSvision

Sistem diterapkan pada proses tenun dan terintegrasi dengan mesin tenun. Saat ditemukan cacat pada kain, sistem akan memberhentikan mesin tenun, menyalakan lampu peringatan, dan menampilkan tempat cacat pada layar [4].

Perkembangan teknologi *artificial intelligence* memunculkan inovasi untuk menerapkan implementasi model AI pada proses industri, khususnya deteksi cacat pada kain. Berbagai riset dan penelitian menggunakan *deep learning* di bidang *computer vision* untuk mendeteksi keberadaan cacat pada berbagai jenis kain.

Pada 2012, Malek [5] melakukan eksperimen untuk mendeteksi cacat pada kain secara otomatis dalam proses industri tekstil dengan menggunakan kamera. Pada 2018, Zhang dkk. [6] menggunakan YOLO-v2 untuk mendeteksi cacat pada 276 gambar kain. Tahun 2019, Li dkk. [7] mendesain sebuah *compact network* sebagai detektor cacat pada kain dalam proses industri dengan ukuran model yang kecil. Guan dkk. [8] menggunakan model VGG untuk mengklasifikasi cacat pada kain. Wei dkk. [9] menggunakan Faster R-CNN berbasis VGG untuk dapat mendeteksi cacat pada kain. Silvestre-Blanes dkk. [10] membuat dataset AITEX dengan menggunakan

line-scan kamera dan pembuatan aplikasi untuk anotasi cacat pada kain. Pada 2020, Peng dkk. [11] membuat PRAN-Net yang teroptimasi untuk mendeteksi cacat pada kain berdasarkan berbagai jenis ukuran cacatnya yang diperoleh dari *bounding box* dari dataset.

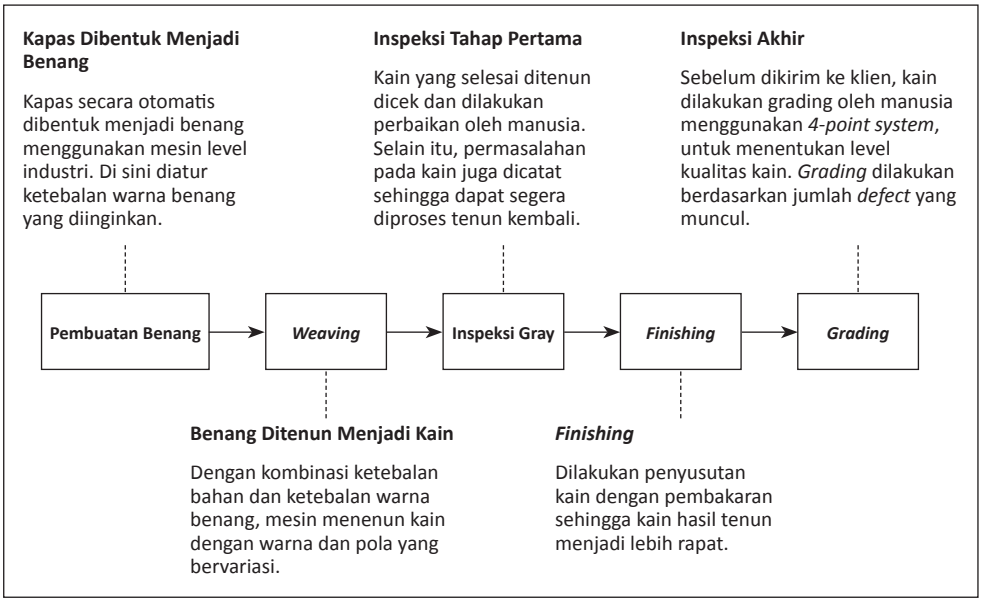
C. METODOLOGI PENELITIAN

Tahapan yang dilakukan untuk melakukan penelitian ini sebagai berikut.

1. Identifikasi Masalah

Untuk mengetahui masalah pada industri tekstil pada pabrik Sinaran Denim, perlu diketahui terlebih dahulu proses produksi tekstil di sana. Gambar 1 adalah diagram yang menjelaskan proses produksi tekstil di pabrik Sinaran.

Proses pertama dalam produksi pabrik adalah pembuatan benang. Kapas secara otomatis dibentuk menjadi benang menggunakan mesin standar industri. Pada proses inilah ketebalan dan warna benang diatur sesuai keinginan. Kemudian, proses pembuatan benang dilanjutkan ke gudang mesin penenun. Mesin dapat menenun benang dengan kombinasi ketebalan bahan dan warna sesuai dengan pesanan pelanggan. Kain denim yang dihasilkan lalu diperiksa dan dilakukan perbaikan cacat manual oleh operator. Cacat yang terdapat pada kain mengindikasikan masalah pada mesin tenun sehingga dibuatkan catatan cacat oleh operator untuk perbaikan mesin tenun. Setelah selesai pemeriksaan, dilakukan pembakaran pada kain sehingga menyebabkan rongga benang pada kain merapat dan ukuran kain menyusut. Tahap

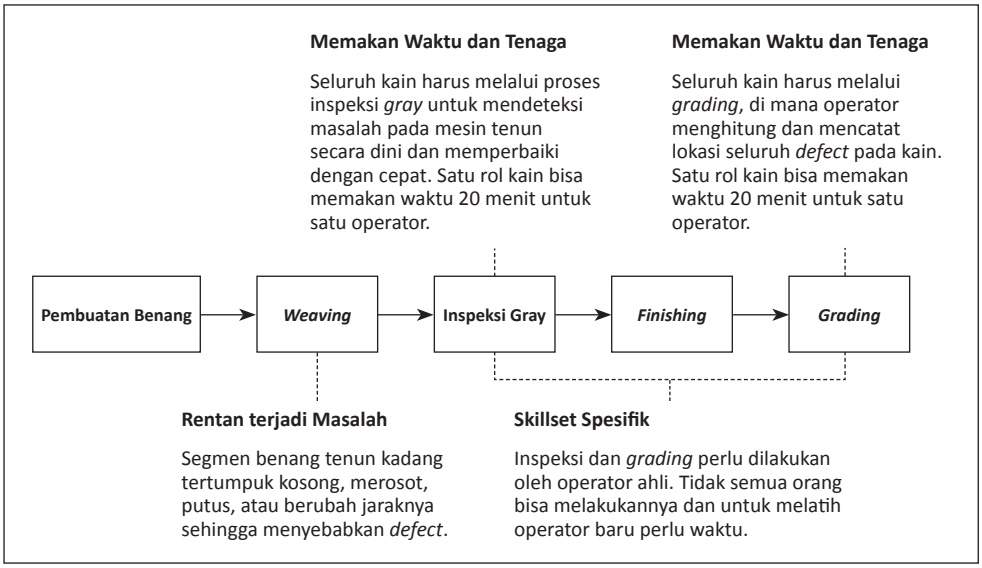


Gambar 1. Proses Produksi Tekstil

ini disebut juga sebagai *finishing*. Hasil dari tahap *finishing* dilanjutkan ke ruang inspeksi *finishing* untuk diperiksa manual seperti sebelumnya. Tujuan pemeriksaan ini adalah menentukan *grade* dari kain dan siap untuk diantar ke konsumen.

Dalam beberapa proses produksi tekstil di pabrik Sinaran Denim, terdapat beberapa tantangan. Gambar 2 menjelaskan tantangan yang dihadapi dari proses produksi tekstil di Sinaran Denim.

Pada mesin tenun sering kali terjadi kesalahan. Saat penenunan, segmen benang tenun kadang bertumpuk, kosong, merosot, putus, atau berubah-ubah jaraknya sehingga menyebabkan cacat pada kain. Selanjutnya, pada proses inspeksi *gray* dibutuhkan waktu 20 menit untuk setiap gulungan kain, sedangkan dibutuhkan proses yang cepat sehingga mesin tenun dapat diperbaiki dan tidak menghasilkan cacat yang lebih banyak lagi. Hal ini juga mirip dengan proses *grading* yang memerlukan waktu 20 menit untuk setiap operator menghitung dan mencatat lokasi seluruh cacat pada kain. Kedua inspeksi ini juga membutuhkan keahlian khusus untuk mengidentifikasi cacat pada kain. Proses-proses yang memiliki tantangan yang menjadi objek penelitian untuk diselesaikan.

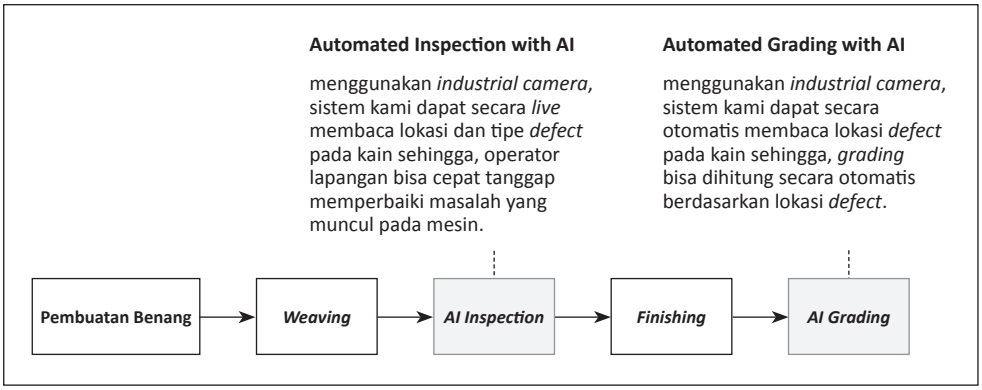


Gambar 2. Tantangan dalam Proses Produksi Tekstil

2. Identifikasi Solusi

Tantangan-tantangan pada proses produksi tekstil dapat diselesaikan dengan *computer vision* yang merupakan bagian dari kecerdasaan artifisial. Kecerdasan artifisial dapat diimplementasi pada proses inspeksi pada pabrik (Gambar 3). Pada inspeksi *gray*, dengan menggunakan *industrial grade camera*, sistem akan membaca secara *live* lokasi dan tipe cacat pada kain. Hal ini memungkinkan operator bisa dengan cepat memperbaiki masalah yang terjadi pada mesin tenun.

Pada inspeksi *finishing* atau *grading*, serupa dengan alat yang dipasang pada inspeksi *gray*, sistem dapat secara otomatis membaca lokasi cacat pada kain dan dapat menentukan *grade* secara otomatis berdasarkan panjang dan lokasi cacat.



Gambar 3. Identifikasi Solusi dengan Kecerdasan Artifisial

3. Rancangan Komponen Sistem

Berikut ini komponen sistem yang harus ada dalam pembuatan sistem deteksi cacat pada tekstil.

a. Infrastruktur data

   Membangun lingkungan penangkapan gambar kain dan penyimpanan.

b. Infrastruktur riset

   Membangun lingkungan untuk *training* model *machine learning*.

c. Sistem deteksi cacat dan aplikasi interaktif

   Mengembangkan sistem untuk penggunaan model hasil *training* dan integrasi dengan aplikasi interaktif.

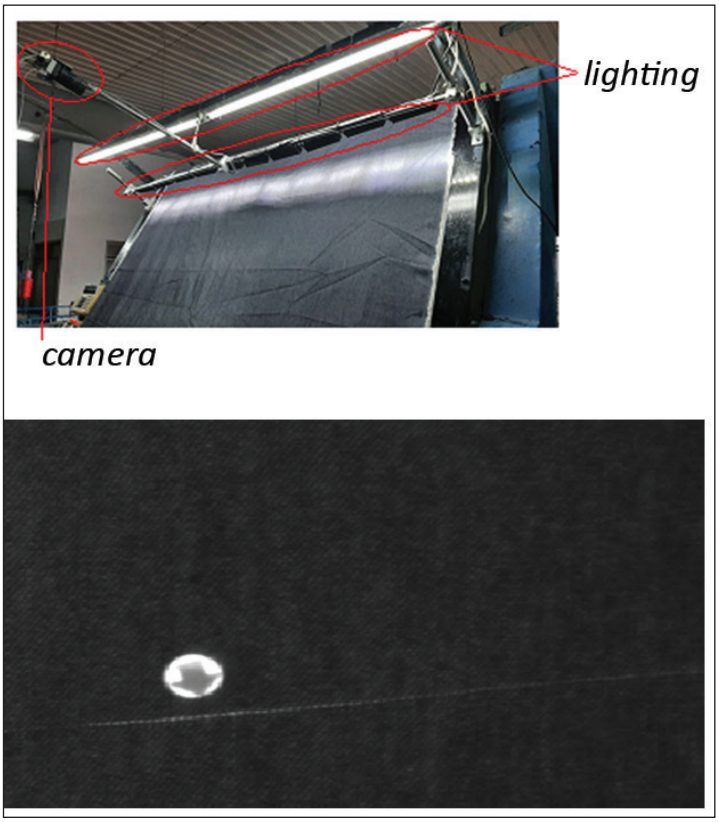
d. Infrastruktur *deployment*

   Menyiapkan infrastruktur untuk menjalankan sistem di pabrik.

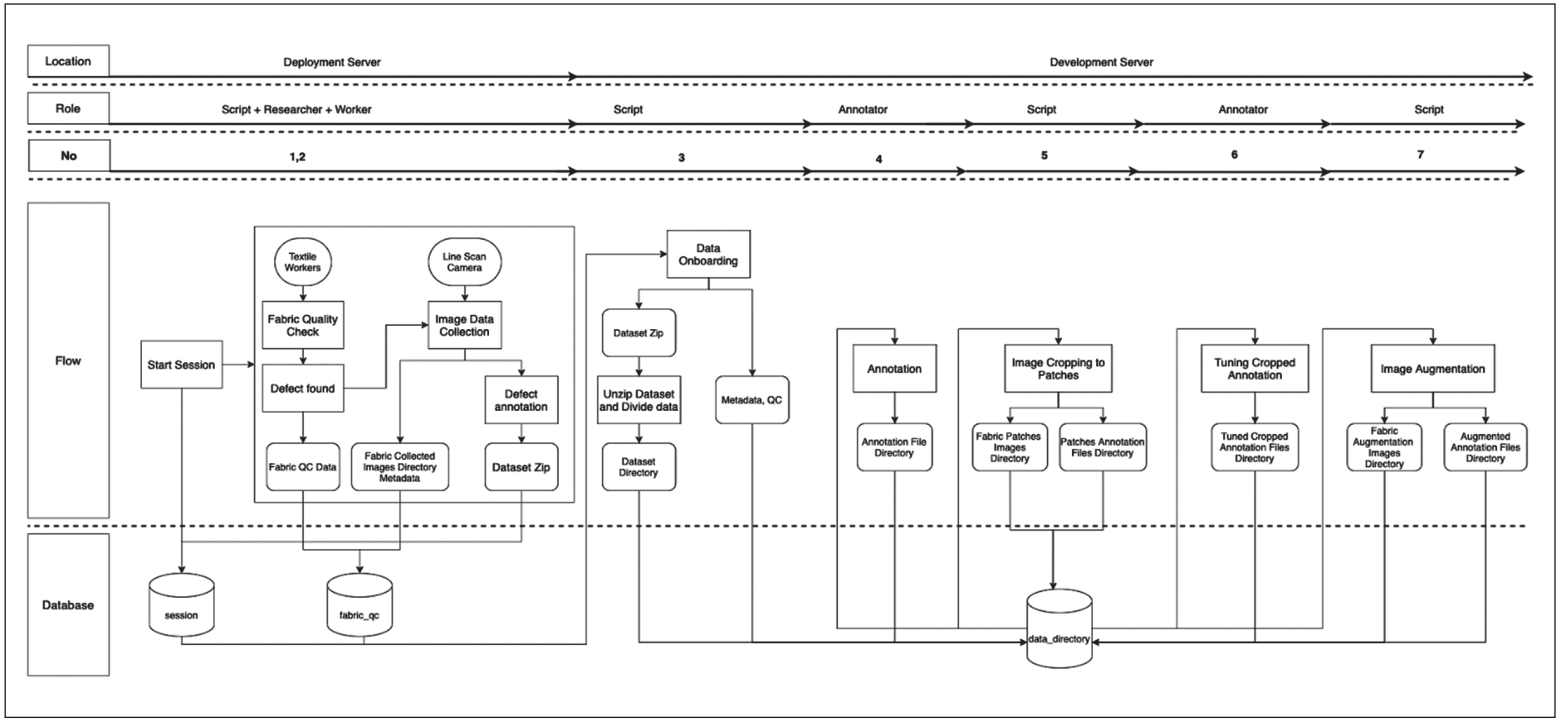
4. Rancangan Alur Pengumpulan dan Persiapan Dataset

Pada mesin inspeksi *gray* di pabrik, dipasang *line scan camera*. Kamera ini digunakan untuk mengumpulkan data dan nanti akan digunakan sebagai pendeteksi cacat juga. Alur dataset diawali dengan operator akan memeriksa kain dengan biasanya didampingi dengan tim riset. Saat cacat ditemukan, operator akan memberitahu tim riset untuk mengambil gambar dengan kamera dan langsung akan melakukan anotasi cacat sesuai dengan jenis cacatnya (Gambar 4). Setelah satu rol kain selesai diperiksa, tim riset akan meminta catatan cacat dan melakukan kompresi zip pada gambar yang telah dikumpulkan, lalu disimpan pada *database*.

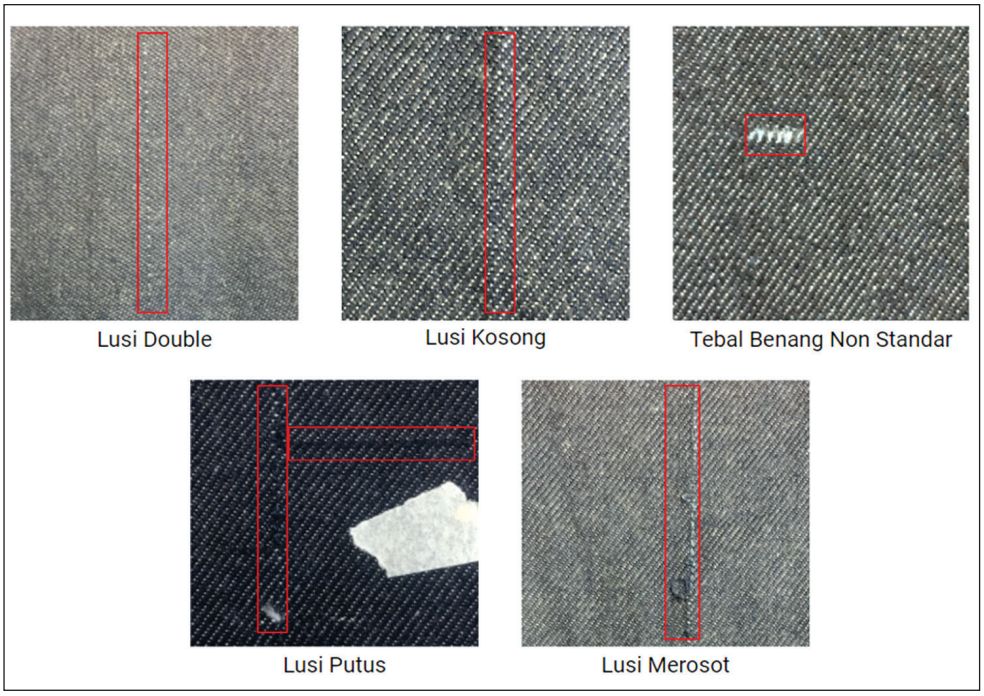
Setelah itu, akan ada *script* yang secara otomatis melakukan transfer data gambar dari *database* pabrik ke *database* riset, lalu *script* akan melakukan *cropping* gambar ke beberapa ukuran sesuai dengan input yang ingin dilakukan *training*. Setelah itu, akan dilakukan *tuning* anotasi untuk gambar yang sudah di-*crop.* Terakhir, akan dilakukan augmentasi pada gambar untuk menambah jumlah dataset. Gambar 5 menjelaskan alur dari data yang didapat beserta keterangan aktor yang akan bertanggung jawab atas data pada proses tertentu.



Gambar 4. Hasil Gambar *Line Scan Camera* pada Kain Denim



Gambar 5. Hasil Gambar *Line Scan Camera* pada Kain Denim



Gambar 6. Contoh Anotasi Cacat dengan Kelasnya

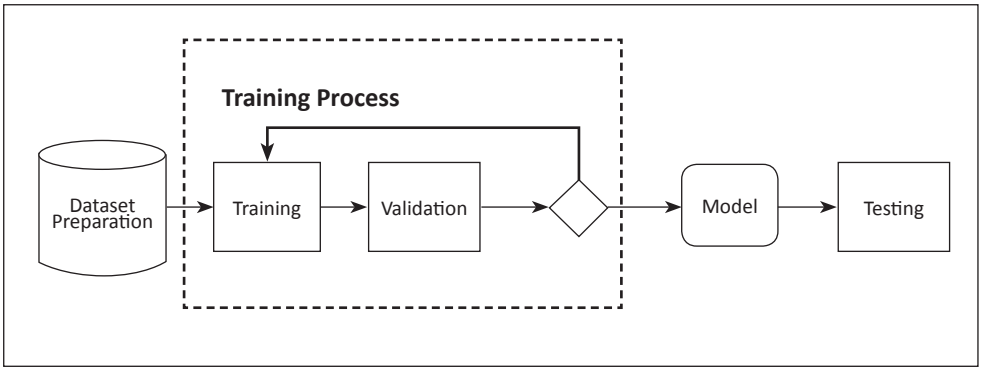
5. Alur Eksperimen *Machine Learning*

Cacat pada kain dideteksi dengan memasukkan input gambar kain ke sebuah model *machine learning* yang telah di-*training* dengan dataset gambar kain. Model *machine learning* yang dipakai menggunakan teknologi berbasis *convolutional neural network* (CNN) yang berfungsi mendapatkan fitur dari gambar atau citra pada kain dengan mudah. *Neural network* (NN) merupakan pemodelan dari cara kerja otak manusia dan NN itu sendiri adalah kumpulan *layers* yang saling terhubung menyerupai sebuah neuron yang keluarannya terhubung ke fungsi non-linear. Kumpulan dari NN yang tersusun banyak disebut *deep neural network* (DNN), dan supaya DNN bisa mempelajari dataset, metode yang digunakan disebut *deep learning* (DL).

Jenis-jenis model *deep learning,* seperti *classification model*, *object detection*, dan *image segmentation* bisa digunakan untuk mendeteksi cacat pada kain. *Classification model* digunakan untuk menentukan apakah sebuah gambar kain memiliki cacat atau tidak. *Object detection model* digunakan untuk menentukan lokasi *bounding box* dari cacat yang ada pada kain, sementara *image segmentation* model digunakan untuk memperoleh lokasi detail dari cacat itu sendiri pada sebuah kontur yang tertutup di sebuah gambar.

Hingga kini, banyak sekali model *deep learning* berbasis CNN yang bisa diadopsi untuk kebutuhan mandiri. Hal ini karena *training* model *deep learning* membutuhkan dataset yang sangat banyak dan komputasi yang sangat berat. Pretrain *Model deep learning* atau *feature extractor* pada model tersebut digunakan untuk mempelajari dataset yang jumlahnya relatif sedikit. Langkah selanjutnya adalah menggunakan pretrain model ini sebagai modal awal untuk melakukan *training* model *deep learning* untuk mendeteksi cacat pada kain.

Adapun proses *training* sebuah model *deep learning* ditunjukkan pada Gambar 7.



Gambar 7. Alur Eksperimen Pembuatan Model *Machine Learning*

a. *Dataset preparation*

Tahap ini mempersiapkan dataset berupa *file* gambar foto kain. Dataset dikategorikan berdasarkan tipe *defect* atau cacat yang ada pada kain. Dataset yang ada kemudian dibagi menjadi *training set, validation set,* dan *test set* dengan rasio 6:2:2 dari keseluruhan dataset telah disiapkan.

b. *Training*

Tahap ini untuk membuat model mempelajari *training set* yang telah disiapkan sebelumnya. Selanjutnya, ditentukan *hyperparameter training model* (*optimizer*, *batches*, *epoch number*). *Training set* dibagi menjadi *batches* yang akan dipelajari oleh model. Model yang telah melihat *batch dataset* akan menghasilkan *loss*. Besarnya *loss* akan memengaruhi sensitivitas pada model untuk mengubah parameter di dalam model itu sendiri. Setelah model melihat *batch* dari *training set*, model akan melihat *batch* dari *validation set*. Ketika model sudah mempelajari seluruh *batch*, maka model sudah mempelajari sebesar 1 *epoch*. Iterasi *training* model terus dilakukan hingga beberapa *epoch*.

Model yang diinginkan saat tahap akhir *training* menghasilkan *loss* yang sangat rendah dan akurasi yang setinggi mungkin. Hal ini ditandai dengan *loss* yang nilainya konvergen.

c. *Validation*

Model yang sedang di-*training* akan dikendalikan oleh parameter yang dihasilkan dari proses *validation*. Proses ini menghasilkan *validation loss* dan *validation accuracy.* Namun, model tidak akan mengubah parameter yang terkandung di dalamnya ketika diberikan input *validation set*. Keabsahan performa sebuah model *deep learning* yang baik dapat diketahui apabila *training loss* dan *validation loss* semakin menurun. Apabila proses *training* diteruskan dan *validation loss* mulai meningkat (walaupun *training loss* tetap menurun), proses *training* harus dihentikan karena model mengalami *overfitting*. Model terbaik dengan akurasi terbaik biasa diambil pada iterasi sesaat sebelum terjadi *overfitting*.

d. *Model*

Model terbaik yang dihasilkan dari *training process* disimpan dalam bentuk sebuah *file. File* ini digunakan untuk dites pada *test set* atau di-*deploy* pada tahap *production*.

e. *Test*

Model yang sudah dilatih bisa digunakan untuk dicobakan pada *test set* untuk melihat performa dari model tersebut.

6. Dataset untuk *Training* Model *Machine Learning*

Dataset diperoleh menggunakan kamera Samsung S21 dengan pengaturan resolusi 8px/mm kain. Diperoleh gambar dengan resolusi 12000x8000 px pada kain dengan lebar 1,5 meter. Gambar kain kemudian di-*crop* dengan ukuran 500x500 px.

Adapun dataset yang digunakan terdiri dari kelas *defect* dan *non-defect*. Kelas *defect* terdiri dari beberapa komponen yang bisa diturunkan menjadi *class* baru, yaitu *horizontal deep stripe, horizontal subtle stripe, horizontal white stripe, vertical deep stripe, vertical subtle stripe, vertical white stripe, contaminated,* dan *messy*.

Tabel 1. Eksperimen Model *Deep Learning* untuk Mendeteksi *Defect* pada Kain

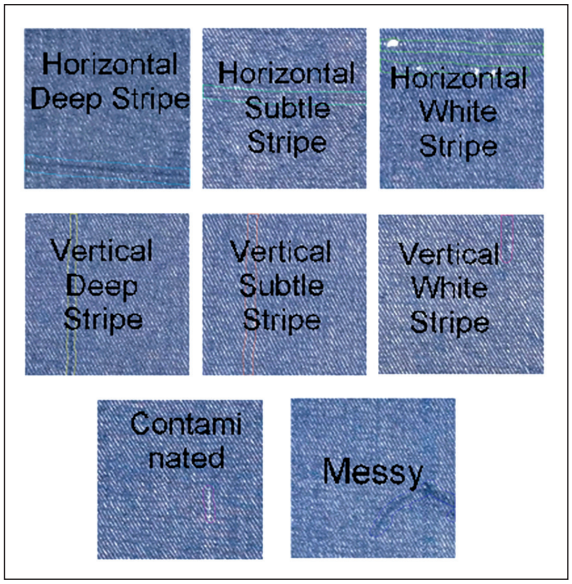
No.	Tipe Model	Class Defect
1	*Classification*	*Defect, Non-Defect*
2	*Detection*	*Defect*
3	*Detection*	*Horizontal Deep Stripe, Horizontal Subtle Stripe, Horizontal White Stripe, Vertical Deep Stripe, Vertical Subtle Stripe, Vertical White Stripe, Contaminated, Messy*
4	*Segmentation*	*Defect*
5	*Segmentation*	*Horizontal Deep Stripe, Horizontal Subtle Stripe, Horizontal White Stripe, Vertical Deep Stripe, Vertical Subtle Stripe, Vertical White Stripe, Contaminated, Messy*

Penentuan *class* ini berdasarkan pola dari *defect* yang muncul terhadap mata orang awam. Tujuannya adalah mempermudah analisis hasil *test set* sehingga bisa dilakukan pengembangan pada eksperimen-eksperimen berikutnya.

Tabel 2. Komposisi Jumlah Dataset pada Setiap *Class*

Class_Training set	*Validation set**	*Test set**	
Defect	282	75	75
Non Defect	282	79	79
Horizontal Deep Stripe	10	5	5
Horizontal Subtle Stripe	94	23	23
Horizontal White Stripe	122	39	39
Vertical Deep Stripe	48	13	13
Vertical Subtle Stripe	13	3	3
Vertical White Stripe	9	2	2
Contaminated	7	2	2
Messy	11	4	4

*) *Validation set* dan *test set* adalah dataset yang sama



Gambar 8. Contoh Tampakan Cacat pada Kain untuk Setiap Kelas

Jumlah dataset yang sedikit dimitigasi dengan metode augmentasi data. Metode augmentasi yang dilakukan berupa rotasi dataset dengan sudut 45, 135, 225, dan 315 derajat.

7. *Evaluation Metrics*

Model kecerdasan artifisial akan mengeluarkan performa yang berbeda pada setiap *class defect*. Pada klasifikasi model *(model classification)*, gambar kain cacat yang terdeteksi cacat disebut *true positive* (TP), sementara apabila tidak terdeteksi cacat disebut dengan *false negative* (FN). Gambar kain yang tidak memiliki cacat apabila terdeteksi cacat disebut *false positive* (FP), sementara apabila tidak terdeteksi cacat disebut *true negative* (TN). Keempat istilah ini dapat dipetakan ke dalam sebuah *confusion matrix* berikut ini.

Tabel 3. *Confusion Matrix* untuk Kasus Cacat pada Kain.

	Cacat	**Tidak Cacat**
Cacat	TP	FN
Tidak Cacat	FP	TN

Kemampuan model untuk memperoleh jumlah kain yang cacat dari semua prediksi kain yang dianggap cacat disebut *precision*.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Apabila *precision* semakin tinggi, semakin kecil kemungkinan model kecerdasan artifisial untuk memprediksi cacat pada gambar kain yang tidak ada cacat. Sementara itu, kemampuan model untuk memprediksi gambar yang cacat dari semua gambar yang mengandung cacat disebut *Recall*.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Apabila *recall* semakin tinggi, semakin kecil kemungkinan model kecerdasan artifisial untuk tidak mendeteksi cacat pada gambar kain yang ada cacat.

Precision dan *Recall* penting untuk menentukan apakah dataset yang diperlukan masih kurang jumlahnya atau kualitas model dalam mendapatkan fitur cacat pada dataset sudah baik atau belum.

Nilai *Precision* dan *Recall* saling memengaruhi satu sama lain. Penyebabnya adalah bagaimana menentukan batas *confidence level/score threshold* dari output model *classification. Threshold* yang rendah menyebabkan semua kemungkinan cacat akan dimunculkan oleh model. Hal ini dapat menyebabkan *false positive* yang tinggi dan menurunkan *precision* karena noise dari hasil prediksi yang tinggi. *Threshold* yang tinggi menyebabkan *false negative* yang tinggi dan menurunkan nilai *recall* karena keluaran model *classification* harus memiliki keyakinan yang tinggi.

Oleh karena itu, model yang baik memiliki nilai *Precision* dan *Recall* yang seimbang dan digabung dalam bentuk F1-score.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Pada kasus model *object detection* dan *image segmentation*, terdapat parameter area yang menjadi indikator apakah model dapat memprediksi dengan benar atau tidak. Walau sebuah gambar memiliki indikasi terdapat cacat, apabila model deteksi tidak dapat menentukan lokasi dari cacat tersebut maka akan dianggap sebagai tidak ada cacat. Oleh karena itu dibutuhkan metrik lain untuk dapat menentukan *F1-score* dari *model detection*. Metrik yang digunakan adalah *intersection over union* (IoU).

$$IoU = \frac{\text{Prediction Area} \cap \text{Ground Truth Area}}{\text{Prediction Area} \cup \text{Ground Truth Area}}$$

Prediction area adalah luas dari hasil prediksi model kecerdasan artifisial, sementara *ground truth* area adalah luas dari hasil anotasi dataset. Sebagai pengaturan umum, nilai IoU yang digunakan adalah 0,5. Nilai ini dapat dianggap sebagai batas (*threshold*). Apabila nilai hasil prediksi mempunyai IoU > 0,5 maka hasil prediksi bisa dinyatakan sebagai *true positive*. Apabila nilai IoU < 0,5 maka hasil prediksi dinyatakan sebagai *false positive*. Apabila model tidak memprediksi cacat, pada gambar yang ada area cacat maka bisa disebut *false negative*.

Apabila sudah menentukan batas IoU, dapat diperoleh diagram relasi dari *precision* dan *recall* dari berbagai score threshold yang disebut PR AUC (*area under curve*). Diagram ini bertujuan menghitung luas area dari PR AUC. Luas ini akan menghasilkan nilai *average precision* (AP) yang berguna sebagai perbandingan performa antara model kecerdasan artifisial yang dibuat secara mandiri dan model *open source*.

8. Aplikasi Pembantu Eksperimen

Aplikasi ini merupakan wadah untuk peneliti dalam melakukan eksperimen *machine learning*, di mana tiap eksperimen merupakan hasil proses *inferencing* dari satu model *machine learning* menggunakan satu *test dataset*. Dari tiap eksperimen, peneliti dapat melihat hasil prediksi model serta nilai evaluasinya. Eksperimen yang telah dilakukan dapat disimpan ke dalam basis data sehingga peneliti dapat melihat riwayat eksperimen yang telah dilakukan.

Agar dapat melihat riwayat eksperimen, aplikasi ini menggunakan basis data sebagai media penyimpanan. Tabel-tabel yang digunakan dalam basis data sebagai berikut.

a. *Data Version*

Tabel ini berisi daftar dataset yang digunakan dalam pengembangan model. Dataset yang terdaftar dalam tabel ini diisi melalui fitur *data preparation*. Dataset yang tersimpan dapat digunakan untuk proses *training model machine learning* atau eksperimen.

b. *Model Source*

Tabel ini berisi daftar jenis model yang digunakan. Jenis model yang terdaftar dalam tabel ini disimpan melalui fitur *model registration*. Data ini digunakan untuk mendaftarkan model hasil *training*.

c. *Model Version*

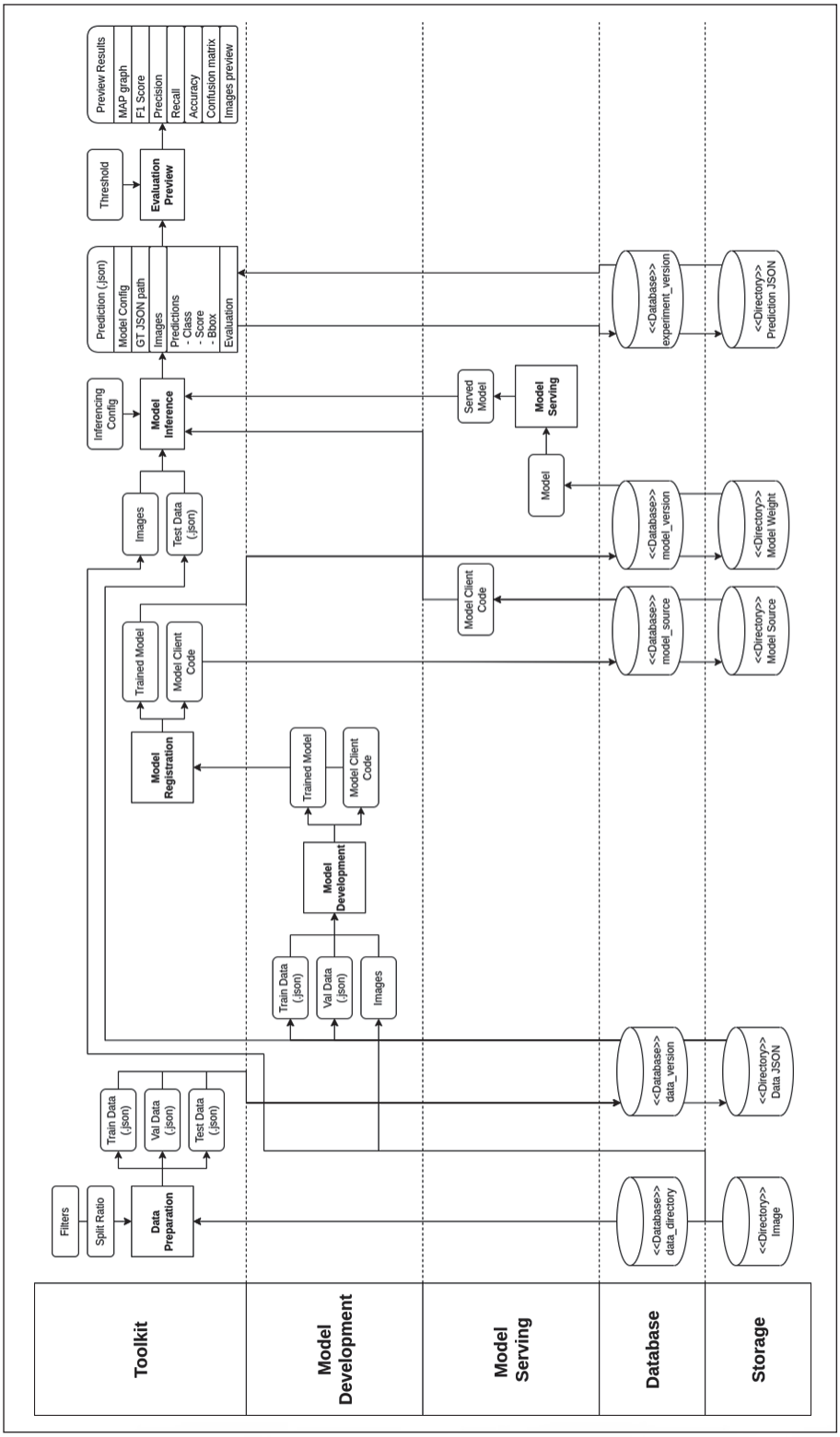
Tabel ini berisi daftar model hasil *training*. Model yang terdaftar dalam tabel ini disimpan melalui fitur *model registration*. Model yang tersimpan dapat digunakan untuk melakukan eksperimen.

d. *Experiment Version*

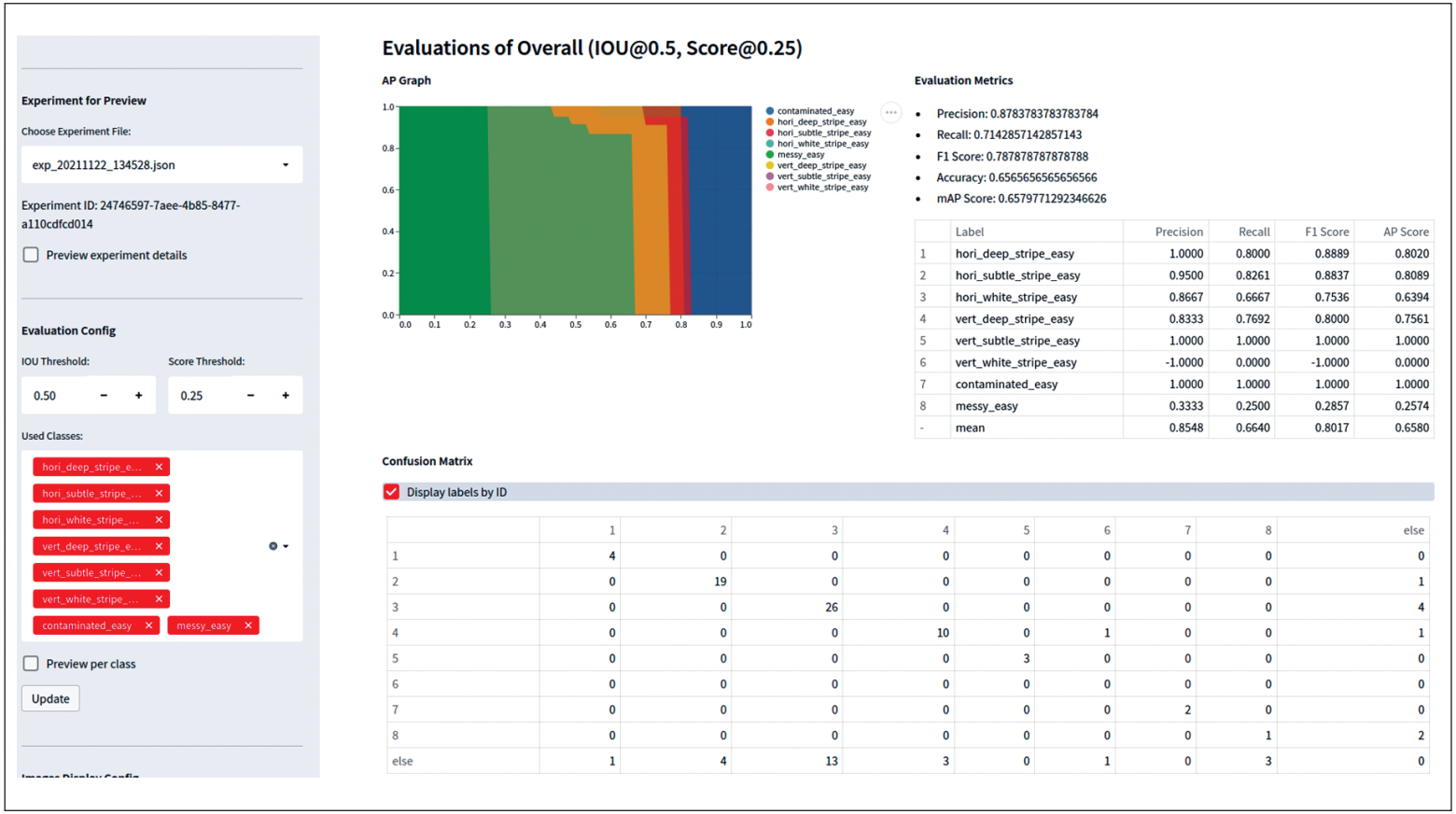
Tabel ini berisi daftar eksperimen yang sudah dilakukan. Data eksperimen dalam tabel ini disimpan melalui fitur *model inference*. Tabel ini dapat digunakan sebagai log untuk melihat riwayat eksperimen.

Berikut adalah fitur yang disediakan oleh aplikasi ini untuk mendukung berjalannya proses eksperimen.

1) *Data preparation*
   Fitur ini berfungsi untuk mempersiapkan dataset dari gambar-gambar yang sudah dikumpulkan beserta anotasinya. Dataset dari fitur ini dapat digunakan dalam pengembangan model, baik dalam proses *training* maupun eksperimen. Dataset yang disiapkan dapat disimpan ke dalam tabel *data version* dan *file*nya dapat disimpan ke dalam *storage* dalam format COCO json. Dataset yang sudah terdaftar dapat digunakan dalam fitur-fitur selanjutnya.
2) *Distribution exploration*
   Fitur ini berfungsi untuk memperlihatkan distribusi dari atribut-atribut yang ada di dalam dataset, seperti jenis label, jumlah anotasi, dan dimensi. Distribusi ditampilkan dalam bentuk grafik histogram. Dari fitur ini, peneliti dapat mengukur seberapa *balance* dataset yang terdaftar.
3) *Image exploration*
   Fitur ini berfungsi untuk memperlihatkan gambar-gambar yang ada di dalam dataset beserta anotasinya.
4) *Model registration*
   Fitur ini berfungsi untuk mendaftarkan model *machine learning* hasil proses training. Model yang didaftarkan akan disimpan ke dalam tabel *model version* dan file modelnya akan disimpan ke dalam *storage*. Model yang sudah terdaftar dapat digunakan untuk eksperimen.
5) *Model inference*
   Fitur ini berfungsi untuk melakukan eksperimen atau proses *inferencing*. Tiap proses *inferencing* dilakukan menggunakan satu dataset dari tabel *data version* dan satu model dari tabel *model version*. Hasil eksperimen dapat disimpan ke dalam tabel *experiment version*, dan hasil prediksi dari proses *inferencing* dapat disimpan ke *storage* dalam bentuk json.
6) *Evaluation preview*
   Fitur ini berfungsi untuk menampilkan hasil dari eksperimen yang tersimpan. Adapun hasil yang ditampilkan sebagai berikut.
   a) Hasil evaluasi model *machine learning*, meliputi *precision, recall*, f1 *score, accuracy, confusion matrix, average precision graph*, dan *mean average precision.*
   b) Gambar-gambar dari dataset yang digunakan dalam eksperimen beserta anotasi dan hasil prediksinya.



Gambar 9. Alur Aplikasi Pembantu Eksperimen



	Label	Precision	Recall	F1 Score	AP Score
1	hori_deep_stripe_easy	1.0000	0.8000	0.8889	0.8020
2	hori_subtle_stripe_easy	0.9500	0.8261	0.8837	0.8089
3	hori_white_stripe_easy	0.8667	0.6667	0.7536	0.6394
4	vert_deep_stripe_easy	0.8333	0.7692	0.8000	0.7561
5	vert_subtle_stripe_easy	1.0000	1.0000	1.0000	1.0000
6	vert_white_stripe_easy	-1.0000	0.0000	-1.0000	0.0000
7	contaminated_easy	1.0000	1.0000	1.0000	1.0000
8	messy_easy	0.3333	0.2500	0.2857	0.2574
-	mean	0.8548	0.6640	0.8017	0.6580

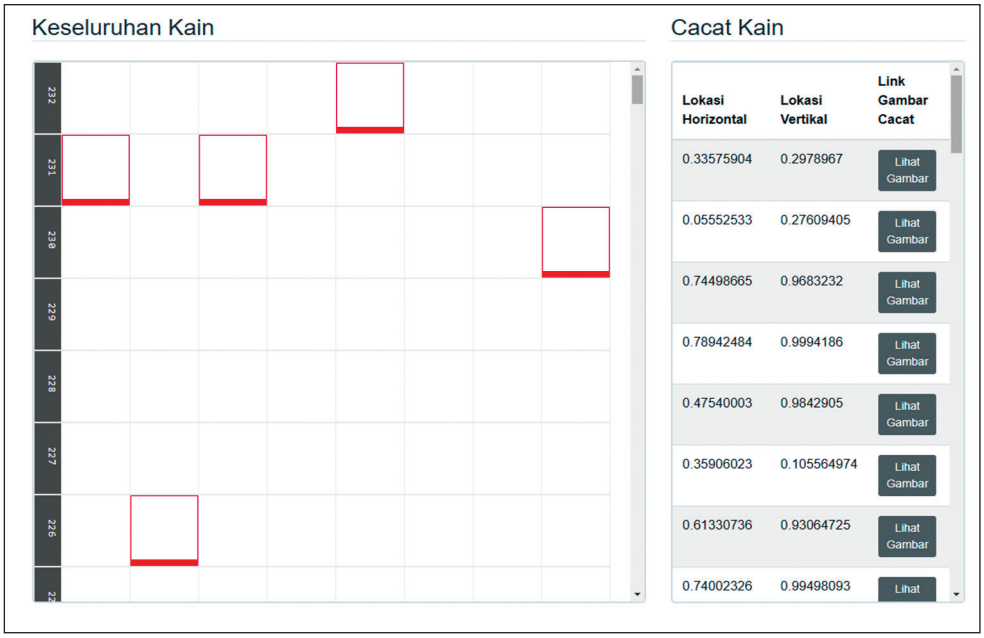
	1	2	3	4	5	6	7	8	else
1	4	0	0	0	0	0	0	0	0
2	0	19	0	0	0	0	0	0	1
3	0	0	26	0	0	0	0	0	4
4	0	0	0	10	0	1	0	0	1
5	0	0	0	0	3	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	2	0	0
8	0	0	0	0	0	0	0	1	2
else	1	4	13	3	0	1	0	3	0

Gambar 10. Contoh hasil evaluasi eksperimen dalam aplikasi

9. Aplikasi *End to End*

a. *Dashboard*

Pada sistem pendeteksi cacat dengan kecerdasan buatan, dibuat *dashboard* agar pengguna dapat mengetahui posisi cacat yang dideteksi lebih mudah. Hal ini diperoleh dengan membuat pemetaan kain dengan gambar di *dashboard* seperti Gambar 11.



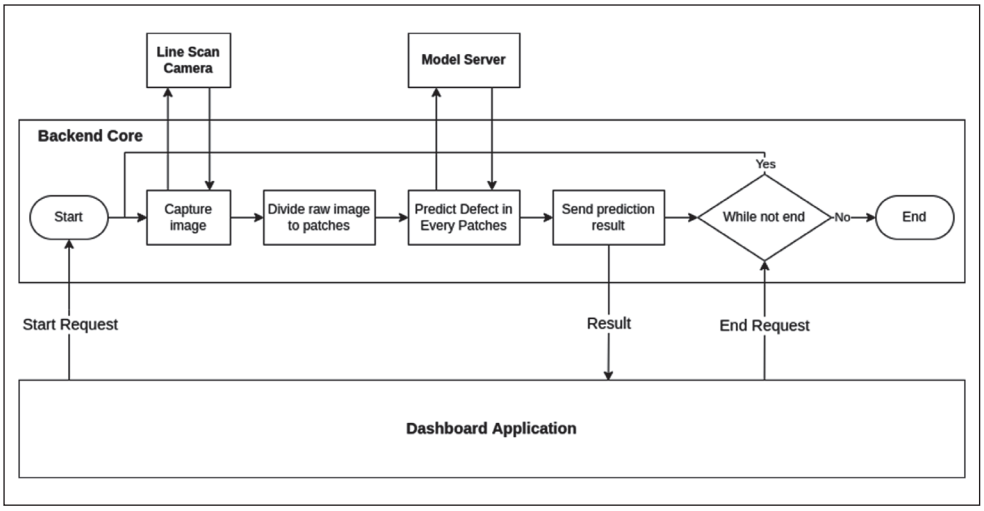
Gambar 11. *Dashboard* Sistem Pendeteksi Cacat

b. *Backend core*

Backend core merupakan *software* yang berfungsi untuk menangkap gambar kain melalui *line scan camera*, kemudian memprosesnya menggunakan model *machine learning* untuk mendapatkan prediksi posisi cacat yang terdapat pada gambar kain. *Software* ini digunakan sebagai program inti dari aplikasi *dashboard*. *Backend core* dan aplikasi *dashboard* saling berkomunikasi melalui *websocket*.

Backend core mulai berjalan ketika menerima *start request*. *Start request* ini dapat dikirimkan dengan menekan tombol "Start" pada aplikasi *dashboard* saat mesin inspeksi kain mulai dijalankan. Ketika berjalan, *backend core* akan secara terus-menerus mengambil gambar kain yang berjalan pada *belt conveyor* melalui *line scan camera*. Tiap gambar yang diambil akan dipecah menjadi beberapa gambar dalam ukuran 500x500 yang disebut sebagai *patch* agar dapat diproses oleh model *machine learning*. Kemudian, tiap patch akan dikirimkan ke model server untuk diproses oleh model *machine learning* dan didapatkan hasil prediksi cacatnya. Model server merupakan suatu modul terpisah yang berfungsi untuk menjalankan model *machine learning*.

Setelah didapatkan hasil prediksinya, *backend core* akan mengirimkan tiap *file patch*, koordinat lokasinya pada kain, serta hasil prediksi cacatnya ke aplikasi *dashboard*. Proses ini akan dilakukan secara terus-menerus hingga *backend core* menerima *end request*. Ketika rol kain habis, pengguna dapat menekan tombol "End" pada aplikasi *dashboard* untuk mengirimkan *end request* dan menghentikan proses pada *backend core*.



Gambar 12. Alur *Backend Core*

D. HASIL

1. Pengaturan eksperimen

Beberapa eksperimen dilakukan dan menghasilkan lima model yang dapat dilaporkan dalam dokumen ini. *Training process* dilakukan di GPU NVIDIA RTX 2080 Ti. *Testing process* dilakukan di GPU NVIDIA RTX 2070 SUPER. Berbagai eksperimen untuk memperoleh model *deep learning* telah dilakukan dengan menggunakan berbagai tipe model dan arsitektur model yang berbeda. Namun, demi kenyamanan, detail dari arsitektur model yang digunakan tidak disebutkan.

2. *Classification model*

a. Pengaturan *Training*

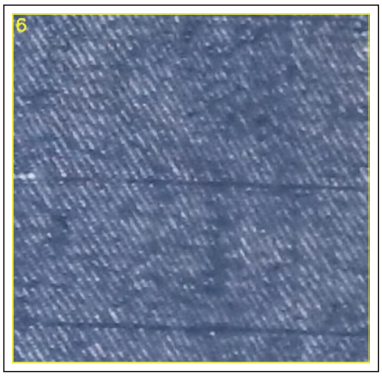
Model classification di-*training* menggunakan *pretrained* model dari arsitektur berbasis CNN untuk dapat melakukan *classification* pada dua kelas (*defect* dan *non-defect*). Model di-*training* dengan *training set* yang berjumlah 564 dan dites pada dataset *test set* yang berjumlah 154. *Training* dijalankan sebanyak 10.000 *step* dan setiap *step* memiliki *batch* 128. Memori yang digunakan di GPU sebesar ~10 GB.

b. Hasil Eksperimen

Metrik utama yang digunakan pada eksperimen ini adalah F1-*score*. Selain F1-*score*, nilai *Precision* (P) dan *Recall* (R) juga dicatat sebagai bahan analisis untuk eksperimen berikutnya. Dari eksperimen yang telah dilakukan, hasilnya sebagai berikut.

Tabel 4. Performa dari *Model Classification* pada Dataset *Defect* dan *Non-Defect*.

Nama Model	Class	Dataset		P	R	F1 Score
		Train	Test			
Classification-01	Defect	282	75	0,70	0,84	0,76
	Non Defect	282	79	0,81	0,67	0,73



Gambar 13. Contoh *True Positive* Model Classification-01

Saat iterasi *test-set*, *model classification* pada Tabel 4 memiliki kecepatan *inference* 20,27 ms/image.

3. *Object Detection Model*

a. Pengaturan Training

Model object detection di-*training* menggunakan *pretrained* model dari arsitektur berbasis CNN single *stage detector* untuk dapat menentukan lokasi pada *single* kelas (*defect*) dan juga pada delapan kelas *defect*. Model di-*training* dengan *training set* yang berjumlah 282 dan dites pada dataset *test set* yang berjumlah 75. *Training* dijalankan sebanyak 10.000 step dan setiap step memiliki *batch* 64. Memori yang digunakan di GPU sebesar ~10 GB

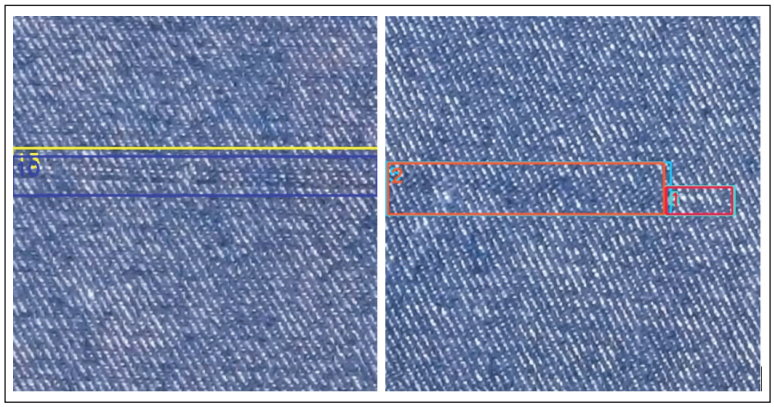
b. Hasil Eksperimen

Metrik utama yang digunakan pada eksperimen ini adalah F1-*score*. Namun, untuk mendapatkan F1-*score*, sebelumnya harus ditentukan nilai *threshold* dari *output* model (IoU *threshold, score threshold*). IoU *threshold* yang digunakan adalah 0,5, sementara

score threshold yang digunakan adalah 0,25. Dari kedua *threshold* ini, didapatkan diagram *Precision* dan *Recall*, beserta *Average Precision* (AP) di IoU=0.5 atau bisa disebut AP50. Hasil eksperimen sebagai berikut:

Tabel 5. Performa dari *Model Detection* pada Dataset *Defect* dan delapan *Class of Defect*

Nama Model	Class	Dataset		P	R	AP_{50}	F1 Score
		Train	Test				
Detection-01	Defect	282	75	0,89	0,69	0,67	0,78
Detection-02	8 Class of Defect	282	75	0,87	0,71	0,65	0,79



Gambar 14. Contoh *True Positive* model Detection-01 dan 02

Saat iterasi *test-set*, model *Detection* di tabel tersebut memiliki kecepatan *inference* 35.74 ms/image.

4. *Image Segmentation* Model

a. Pengaturan *Training*

Model *image segmentation* di-*training* menggunakan *pretrained* model dari arsitektur berbasis CNN *two-stage detector* untuk dapat menentukan lokasi pada *single* kelas (*defect*) dan juga pada delapan kelas *defect*. Model di-*training* dengan *training set* yang berjumlah 282 dan dites pada dataset *test set* yang berjumlah 75. Adapun model Segmentation-02 di-*training* dengan dataset tambahan dari augmentasi rotasi karena jumlah dataset yang sedikit mengakibatkan performa model yang kurang baik. *Training* dijalankan sebanyak 100 epoch dan setiap *step* memiliki *batch* 2. Memori yang digunakan di GPU sebesar ~5 GB

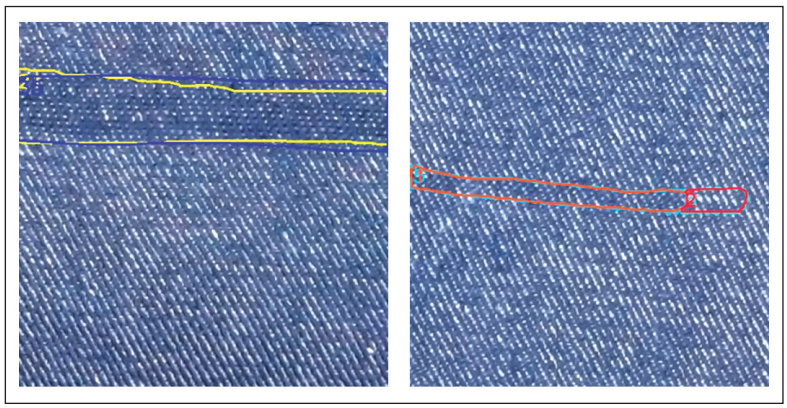
b. Hasil Eksperimen

Metrik utama yang digunakan pada eksperimen ini adalah F1-score. IoU *threshold* yang digunakan adalah 0, 5 dan *score threshold* yang digunakan adalah 0,3. Dari kedua *threshold* ini, didapatkan diagram *Precision* dan *Recall*, beserta AP50.

Tabel 6. Performa dari Model Segmentation pada Dataset Defect dan 8 Class of Defect

Nama Model	Class	Dataset		P	R	AP_{50}	F1 Score
		Train	Test				
Segmentation-01	Defect	282	75	0,87	0,71	0,65	0,79
Segmentation-02	8 Class of Defect	450*	75	0,88	0,64	0,64	0,74

*)Hasil dari augmentasi dataset



Gambar 15. Contoh *True Positive Model Segmentation*-01 dan 02

Saat iterasi *test-set*, model *Segmentation* pada Tabel 6 memiliki kecepatan *inference* 153,26 ms/image.

E. KESIMPULAN

Berdasarkan percobaan yang telah dilakukan, kecerdasan artifisial dapat membuat industri tekstil lebih efisien. Hal ini didapat dengan cara mengotomasi proses inspeksi kain yang ada dengan *machine learning* sehingga operator yang ada dapat dialokasikan untuk pekerjaan lain, mesin penenun dapat diperbaiki dengan segera, dan cacat yang tercipta juga lebih sedikit, bahkan menghemat banyak waktu.

Akan tetapi, performa dari *machine learning* memang harus lebih ditingkatkan lagi sehingga setiap cacat pada industri tekstil dapat terdeteksi semua dan *grade* kain yang ditentukan juga lebih baik lagi. Selain itu, sistem pendukung, seperti infrastruktur dan aplikasi *dashboard*, untuk riset juga harus dibuat dengan baik.

Beberapa hal yang dapat dikembangkan lagi sebagai berikut.

1. Meningkatkan performa pada kain bercorak yang diintegrasikan dengan sistem manajemen pabrik.
2. Mempersiapkan skalabilitas kepada seluruh mesin di pabrik.
3. Memperluas performa sistem pada jenis kain yang beragam.
4. Memperluas *partnership* dengan mitra lain.

UCAPAN TERIMA KASIH

Riset ini dilakukan oleh tim ahli dari Riset AI. Riset AI adalah perusahaan yang berfokus pada riset berbasis kecerdasan artifisial dengan fokus utama pada implementasinya di bidang *computer vision*. Terima kasih banyak kepada Pak Arianto selaku pemilik Sinaran Denim yang sudah bersedia bekerja sama dengan Riset AI. Terima kasih kepada pihak pabrik, yaitu Pak Zainal selaku *Head of Factory*, Pak Sugi, Pak Tono, dan orang-orang yang tidak dapat disebutkan dalam tulisan ini yang telah ikut menyukseskan riset ini.

DAFTAR PUSTAKA

[1] Kemenperin. "Making Indonesia 4.0: Strategi RI masuki revolusi industri Ke-4." Accesssed: Oct 18, 2022. [Online]. Available: https://kemenperin.go.id/artikel/18967/Making-Indonesia-4.0:-Strategi-RI-Masuki-Revolusi-Industri-Ke-4

[2] A. Kristianus. "Kuartal I-2021, Industri Pengolahan Terkontraksi 1,38%." Investor.id. Accessed: Oct 18, 2022. [Onlie] Available: https://investor.id/business/247277/kuartal-i2021-industri-pengolahan-terkontraksi-138

[3] Advantech. "AI Defect Inspection for Textile." Advantech.com. Accessed: December 1, 2021. [Online] Available: https://www.advantech.com/resources/case-study/ai-defect-inspection-for-textile

[4] "Automatic inspection | BMSvision." Bmsvision.com, Accessed: Dec 20, 2021. [Online]. Available: https://www.bmsvision.com/products/automatic-inspection.

[5] A. S. Malek, "Online fabric inspection by image processing technology," Doctoral dissertation, Mulhouse, 2012.

[6] H. -w. Zhang, L. -j. Zhang, P. -f. Li, dan D. Gu, "Yarn-dyed fabric defect detection with YOLOV2 based on deep convolution neural networks," in *IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS) 2018,* pp. 170–174, doi: 10.1109/DDCLS.2018.8516094.

[7] Y. Li, D. Zhang, dan Dh-J Lee, "Automatic fabric defect detection with a wide-and-compact network," *Neurocomputing*, 329, pp. 329–338.

[8] M. Guan, Z. Zhong, Y. Rui, H. Zheng, dan X. Wu, "Defect detection and classification for plain woven fabric based on deep learning," in *2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*, pp. 297–302.

[9] B. Wei, K. Hao, X-S Tang, dan L. Ren, "Fabric defect detection based on faster RCNN," in *International Conference on Artificial Intelligence on Textile and Apparel*, W. Wong (Ed), Okt. 2018, vol. 849, pp. 45–51 https://doi.org/10.1007/978-3-319-99695-0_6

[10] J. Silvestre-Blanes, T. Albero Albero, I Miralles, I., Pérez-Llorens, R., dan J. Moreno, "A public fabric *database* for defect detection methods and results," *Autex Research Journal,* vol. 19, no. 4, pp. 363–374, Juni 2019, doi: 10.2478/aut-2019-0035.

[11] P. Peng, Y. Wang, C. Hao, Z. Zhu, T Liu, dan W. Zhou, "Automatic fabric defect detection method using PRAN-net," *Applied Sciences*, vol. 10, no. 23, p. 8434, Nov. 2020, doi: 10.3390/app10238434.