BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian mengenai deteksi *deepfake* sudah pernah dibuat sebelumnya. Penelitian pertama oleh Karandikar dkk. (2020). Penelitian ini bertujuan untuk mendeteksi *deepfake* menggunakan *Convolutional Neural Network* (CNN) dan *transfer learning*. Pendekatan ini menganalisis setiap *frame* dari video. Model ini dilatih menggunakan *dataset Celeb-DF*, yang terdiri dari 4000 video asli dan 5000 video *deepfake* sebagai data pelatihan. Model yang dihasilkan menunjukkan akurasi prediksi sekitar 70%.

Penelitian kedua dilakukan oleh Jamsheed & Janet (2021). Penelitian ini bertujuan untuk membuat sistem prediksi *deepfake* menggunakan kombinasi *Convolutional Neural Network* (CNN) dan *Long Short Term Memory* (LSTM). CNN digunakan untuk mengekstrak fitur dari setiap *frame*, sedangkan LSTM berfungsi untuk mengklasifikasi video asli dan *deepfake*. *Dataset* yang digunakan meliputi DFDC, *FaceForensic*, *Celeb-DF*, dan yang lainnya. Jumlah *dataset* yang digunakan pada penelitian ini sekitar 5000 data video. Penelitian ini menunjukkan akurasi prediksi sebesar 92%.

Penelitian ketiga dilakukan oleh Zhu dkk. (2021). Penelitian ini bertujuan untuk mendeteksi *deepfake* menggunakan pendekatan dekomposisi. Yang berfokus pada pencarian detail pemalsuan yang tersembunyi terkait cahaya langsung dan tekstur yang tidak konsisten. *Dataset* yang digunakan adalah *FaceForensics++* dengan metode manipulasi *Face2Face* (F2F) dan *FaceSwap* (FS) dengan 1000 data yang digunakan untuk pelatihan. Hasil penelitian ini memperoleh akurasi prediksi sebesar 86% untuk video manipulasi dengan F2F dan 92% untuk video manipulasi dengan FS.

Penelitian keempat dilakukan oleh Abidin dkk. (2022). Penelitian ini bertujuan untuk mendeteksi *deepfake* menggunakan ResNext *Convolutional Neural Network* (CNN) dan *Long Short Term Memory* (LSTM). ResNext CNN digunakan untuk ekstraksi fitur yang ada pada video, sedangkan LSTM digunakan untuk klasifikasi hasil

ekstraksi fitur tersebut. *Dataset* yang digunakan berupa gambar digital dari situs penyedia data yang telah banyak digunakan oleh peneliti, tetapi jumlah spesifik data yang digunakan tidak dijelaskan. Hasil penelitian ini memperoleh akurasi prediksi paling tinggi sebesar 90% pada data video yang dipecah menjadi 60 *frame*.

Penelitian kelima dilakukan oleh Saputro (2023). Penelitian ini bertujuan untuk melakukan deteksi *deepfake* menggunakan metode pendekatan *Triplet Loss* dengan model algoritma *Random Forest* (RF) dan *Stochastic Gradient Descend* (SGD) menggunakan *Multi-task Cascaded Convolutional Network* (MTCNN). MTCNN digunakan untuk ekstrak fitur wajah dari seitap *frame*, kemudian dimasukkin ke dalam model *Triplet Loss* untuk dilakukan pengelompokan fitur yang sama, dan akan diklasifikasi menggunakan SGD dan RF. *Dataset* yang digunakan untuk pelatihan adalah *FaceForensics++* dengan total 600 video. Hasil penelitian ini menunjukkan akurasi prediksi sebesar 86% menggunakan model RF, dan 84% menggunakan model SGD pada jumlah *frame* 30.

Tabel 2.1 Tabel Penelitian Terdahulu

Peneliti	Metode	Data	Jumlah Data	Hasil
(Karandikar dkk., 2020)	*Convolutional Neural Network* (CNN)	*Celeb-DF*	9000	Akurasi sebesar 70%
(Jamsheed dan Janet, 2021)	*Convolutional Neural Network* (CNN) dan *Long Short Term Memory* (LSTM)	DFDC, *FaceForensic*, *Celeb-DF*, dan yang lainnya	1000	Akurasi sebesar 92%
(Zhu dkk., 2021)	Pendekatan Dekomposisi	FaceForensics++ dengan metode manipulasi Face2Face (F2F) dan FaceSwap (FS)	1000	Akurasi sebesar 86% untuk video manipulasi dengan F2F dan 92% untuk video manipulasi dengan FS.

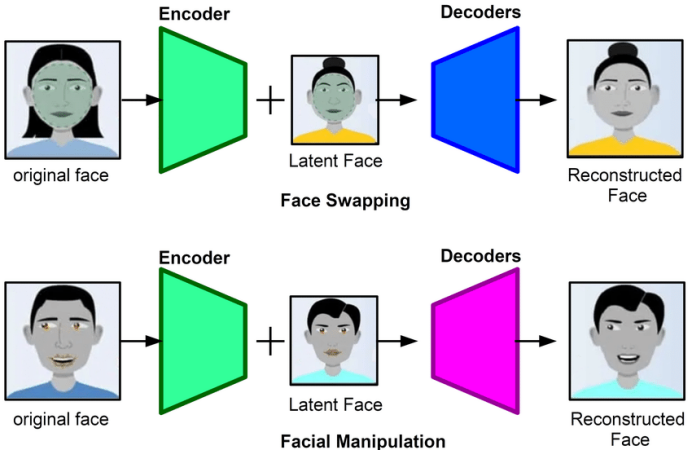
Peneliti	Metode	Data	Jumlah Data	Hasil
(Abidin dkk., 2022)	ResNext *Convolutional Neural Network* (CNN) dan *Long Short Term Memory* (LSTM)	-	-	Akurasi 90% pada data video yang dipecah menjadi 60 *frame*.
(Saputro, 2023)	Pendekatan *Triplet Loss* dengan model algoritma *Random Forest* (RF) dan *Stochastic Gradient Descend* (SGD) menggunakan *Multi-task Cascaded Convolutional Network* (MTCNN).	*FaceForensics++*	600	Akurasi 86% menggunakan model RF, dan 84% menggunakan model SGD pada jumlah *frame* 30.

Berdasarkan Tabel 2.1, terdapat beberapa penelitian yang telah berhasil mengembangkan model deteksi *deepfake* menggunakan metode yang berbeda. Masing-masing metode yang dikembangkan menghasilkan akurasi yang bervariasi, mulai dari yang terendah 70% hingga yang tertinggi 92%. Dataset yang digunakan merupakan data yang telah disediakan khusus untuk pengembangan model terkait deteksi *deepfake*. Dalam penelitian ini, akan melakukan pengembangan model CNN dengan melakukan evaluasi terhadap faktor-faktor yang mempengaruhi akurasi serta implementasi model pada aplikasi Android. Dengan penelitian ini dilakukan, diharapkan dapat mencapai akurasi yang lebih tinggi dibandingkan dengan beberapa penelitian terdahulu.

2.2 Landasan Teori

2.2.1 Deepfake

Deepfake merupakan hasil istilah yang berasar dari gabungan dua kata, yaitu "*Deep Learning*" dan "*Fake*". Ini merujuk pada penggunaan teknologi *deep learning* untuk melakukan pemalsuan atau manipulasi konten. Secara lebih spesifik, *deepfake* memanipulasi elemen-elemen seperti wajah, gerakan, atau suara pada gambar maupun video. Seolah-olah menciptakan ilusi bahwa seseorang sedang melakukan sesuatu yang sebenarnya tidak terjadi. Teknologi *neural network* yang digunakan pada *deepfake* mempelajari dan menganalisis data sampel dengan jumlah yang besar untuk meniru ekspresi, gerakan, atau suara seseorang. Dengan kata lain, *deepfake* memanfaatkan teknologi *facial mapping* dan AI untuk mengubah elemen-elemen visual tersebut dalam sebuah konten visual (Saputro, 2023).



Gambar 2.1 Proses Pergantian Wajah (Yasrab dkk., 2021)

Dari Gambar 2.1, di atas dapat dilihat proses spesifik pergantian wajah. Wajah asli yang akan akan dimanipulasi dimasukkan ke dalam tahap *encoder*, bersama dengan target wajah yang akan diganti. Hasilnya akan masuk ke tahap *decoder* untuk menghasilkan sebuah gambar yang telah dimanipulasi.

2.2.2 Deep Learning

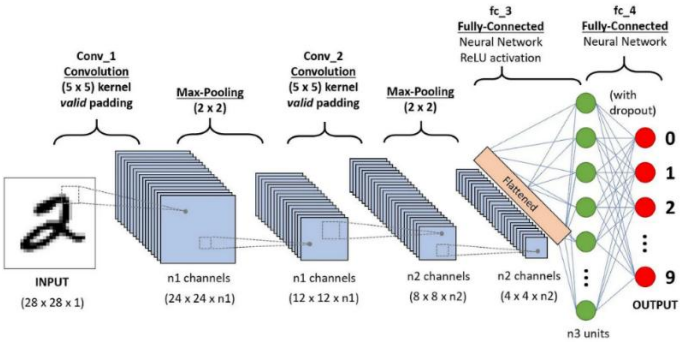
Deep Learning merupakan bidang turunan dari *machine learning* (ML) yang memungkinkan komputer untuk belajar berdasarkan *experience* atau pengalaman dan mengerti dunia dipandang dari segi hierarki suatu konsep. Sehingga deep learning ini akan melakukan pembelajaran dengan sendiri dari pengetahuan yang kita berikan dan menghasilkan sebuah hierarki dalam bentuk lapisan yang banyak dan dalam atau *deep* (Saputro, 2023) (Nugroho dkk., 2020). Ada beberapa jenis algoritma dari *deep learning* yaitu:

1) *Convolutional Neural Network* (CNN)
2) *Recurrent Neural Network* (RNN)
3) *Long Short Term Memory Network* (LTSM)
4) *Self Organizing Maps* (SOM)

2.2.3 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah salah satu algoritma *deep learning* yang banyak digunakan dalam pengolahan data citra, seperti gambar atau video. CNN digunakan untuk menganalisis gambar visual, mendeteksi dan mengenali objek dalam gambar, yang merupakan vektor berdimensi tinggi yang akan melibatkan banyak parameter untuk menggambarkan fitur-fitur jaringan tersebut (Nugroho dkk., 2020). Terdapat beberapa arsitektur CNN, di antaranya LeNet-5, AlexNet, VGG, GoogLeNet, ResNet, SqueezeNet, dan MobileNet (Pamungkas, 2023). Salah satu arsitektur CNN terbaru yang dikembangkan pada tahun 2017 oleh *Microsoft Research*, yaitu ResNeXt, yang merupakan pengembangan dari arsitektur ResNet. ResNeXt menawarkan fleksibilitas dalam mengoptimalkan arsitektur untuk berbagai tugas dan menunjukkan kinerja yang lebih baik dalam hal akurasi klasifikasi (Xie dkk., 2016).

Komponen-komponen utama yang membantu CNN dala mengenali pola dan fitur dalam gambar meliputi *Convolutional Layers*, *Rectified Linear Unit*, *Pooling Layers*, dan *Fully Connected layers (Keita, 2023)*. Untuk penjelasan lebih lanjut, dapat dilihat pada Gambar 2.2.



Gambar 2.2 Komponen dari CNN (Keita, 2023)

Menurut (Keita, 2023), beberapa penjelasan mengenai komponen-komponen dalam CNN adalah sebagai berikut:

1) *Convolutional Layers*

Convolutional Layers merupakan lapisan pertama yang akan dilalui pada CNN, yang memiliki tugas utama yaitu konvolusi. Lapisan ini berfungsi untuk mencari bagian-bagian atau pola tertentu dalam gambar menggunakan *filter* atau *kernel*. Setiap pola atau bagian yang terdeteksi akan diberi tanda atau representasi. Setiap *filter* atau *kernel* pada lapisan ini memiliki fungsi yang berbeda-beda, seperti mendeteksi garis, lengkung, atau bentuk tertentu. Hasil akhir dari konvolusi ini adalah gambar yang direpresentasikan dalam bentuk matriks.

2) *Rectified Linear Unit* (ReLU)

Rectified Linear adalah fungsi aktivasi yang diterapkan pada setiap hasil operasi konvolusi. Fungsi ini membantu jaringan untuk mempelajari hubungan *non-linear* antar fitur-fitur dalam gambar, sehingga meningkatkan kemampuannya dalam mengenali pola-pola yang kompleks. Selain itu, ReLU juga mengurangi masalah *vanishing gradient*, yang sering terjadi pada jaringan yang lebih dalam.

3) *Pooling Layers*

Pooling Layers berfungsi untuk mengambil fitur-fitur penting dari hasil konvolusi. Pada tahap ini, dilakukan beberapa operasi

agregasi yang akan mengurangi dimensi matriks hasil konvolusi, yang dikenal dengan sebutan *feature map*. Pengurangan dimensi ini juga membantu mengurangi penggunaan memori saat pelatihan jaringan. Selain itu, *pooling* juga berfungsi untuk mengurangi *overfitting*. Hasil akhir dari lapisan ini adalah sebuah *feature map* yang lebih kecil, yang kemudian dipersiapkan untuk diproses lebih lanjut oleh *fully connected layers*.

4) *Fully Connected Layers*

Fully connected layers merupakan lapisan terakhir dari CNN. Fungsi ReLU diterapkan pada layer ini untuk memberikan *non-linearity* (ketidaklinearitas). Lapisan ini menghasilkan nilai probabilitas untuk setiap kemungkinan label *output*, dan label yang diprediksi adalah yang memiliki skor probabilitas tertinggi.

2.2.4 Confusion Matrix

Confusion Matrix merupakan perhitungan kinerja algoritma klasifikasi. Beberapa indikator yang biasanya digunakan adalah akurasi, presisi, dan *recall*. *Confusion matrikx* mengandung semua informasi mentah tentang prediksi yang dibuat oleh model klasifikasi pada data yang diberikan. Untuk mengevaluasi akurasi generalisasi model, biasanya digunakan dataset yang tidak digunakan selama proses pembelajaran model.

Tabel 2.2 Tabel Confusion Matrix

		Nilai Prediksi	
		Positif	Negatif
Nilai Sebenarnya	Positif	*True* Positif (TP)	*False* Negatif (FN)
	Negatif	*False* Positif (TP)	*True* Negatif (TN)

Penjelasan dari Tabel 2.2:

1) *True* Positif (TP) merupakan jumlah prediksi positif yang benar
2) *False* Positif (FP) merupakan jumlah prediksi positif yang salah
3) *True* Negatif (TN) merupakan jumlah prediksi negatif yang benar

4) *False* Negatif (FN) merupakan jumlah prediksi negatif yang salah

Dari hasil prediksi yang telah didapatkan, dapat dilakukan perhitungan untuk mencari nilai akurasi, presisi, dan *recall* dengan rumus:

$$Akurasi = \frac{TP+TN}{TP+FP+FN+TN} \quad (1.1)$$

$$Presisi = \frac{TP}{TP+FP} \quad (1.2)$$

$$Recall = \frac{TP}{TP+FN} \quad (1.3)$$

2.2.5 Black Box Testing

Black Box Testing merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dieksekusi pada perangkat lunak dan kemudian keluar dari perangkat lunak dicek apakah telah sesuai yang diharapkan. Dapat diartikan juga sebagai sebuah pengujian yang melakukan pendekatan pengujian untuk mengetahui apakah semua fungsi perangkat lunak telah berjalan semestinya sesuai dengan kebutuhan fungsional yang telah didefinisikan (Fahrezi dkk., 2022).

2.2.6 Android

Android merupakan sistem operasi *open source* yang banyak digunakan pada perangkat *smartphone* ataupun tablet. Android menawarkan kemudahan kepada *developer* aplikasi Android, yang mana *developer* dapat menjalankan aplikasi di berbagai perangkat yang menggunakan sistem operasi Android. Versi Android pertama yakni 1.0 dirilis pada September 2008 oleh Google. Dikarenakan sistem operasinya bersifat open source, maka *smartphone* yang dijual menggunakan sistem operasi ini akan jauh lebih murah dibangding menggunakan sistem operasi yang tidak open source. Awal pengembangan aplikasi Android menggunakan bahasa pemrograman Java. Tetapi pada tahun 2019, Google mengumumkan bahwa *developer* dapat membuat aplikasi menggunakan bahasa pemrograman Kotlin, yang dipercaya lebih ekspresif dan ringkas dan mudah diintegrasikan dibandingkan bahasa Java (*Android Developer*, 2024).