E-ISSN : 3031-9692 Vol. 1 No. 2 2024
Institut Teknologi dan Bisnis STIKOM Bali, 22-04-2024

Pengaruh Optimizer Adam, AdamW, SGD, dan LAMB terhadap Model Vision Transformer pada Klasifikasi Penyakit Paru-paru

Nyoman Sarasuartha Mahajaya[1)], Putu Desiana Wulaning Ayu[2*)], Roy Rudolf Huizen[2)]
Program Studi Magister Sistem Informasi
Institut Teknologi dan Bisnis STIKOM Bali
Denpasar, Indonesia
e-mail: ***222012002@stikom-bali.ac.id[1], wulaning_ayu@stikom-bali.ac.id[2*], roy@stikom-bali.ac.id[2],***

Abstrak

Penyakit paru-paru, sebagai penyebab kematian signifikan secara global, memerlukan diagnosis yang cepat dan akurat untuk mengoptimalkan hasil pengobatan. Penelitian ini menganalisis efektivitas penggunaan berbagai optimizer pada model Vision Transformer untuk klasifikasi penyakit paru-paru. Dalam penelitian ini, dibandingkan empat optimizer diantaranya Adam, AdamW, SGD, dan LAMB, untuk menentukan mana yang paling efektif dan efisien dalam meningkatkan akurasi dan kecepatan pelatihan model. Dengan menggunakan dataset gambar rontgen dada, model dilatih melalui 50 epoch dengan batch size 32 dan learning rate 0.0001. Selain itu, penelitian ini juga membahas pengaruh parameter optimizer terhadap model dalam mengidentifikasi berbagai jenis penyakit paru-paru pada gambar rontgen. Mengukur akurasi, presisi, recall, dan skor F1, serta durasi tiap epoch. Hasil penelitian menunjukkan bahwa optimizer Adam memiliki kinerja tercepat dengan rata-rata waktu 652.71 detik per epoch, sementara LAMB membutuhkan waktu terpanjang dengan rata-rata waktu 689.94 detik per epoch. SGD, meskipun waktu pelatihannya serupa berada diantara Adam dan LAMB, menunjukkan performa yang kurang optimal tetapi dapat menjadi alternatif yang tepat dalam situasi di mana stabilitas lebih penting daripada kecepatan.

Kata kunci: *vision transformer, optimizer, adam, lamb, sgd*

1. Pendahuluan

Diagnosis yang cepat dan tepat dalam dunia kedokteran sangat penting untuk meningkatkan angka kesembuhan dan mengurangi angka kematian, terutama pada penyakit paru-paru, yang merupakan penyebab kematian global yang signifikan. Teknik diagnostik saat ini untuk penyakit paru-paru sering kali kurang sensitif dan cepat, sehingga menyebabkan keterlambatan diagnosis dan perawatan yang tidak efektif [1],[2]. Kecerdasan buatan telah menunjukkan harapan dalam meningkatkan diagnosis penyakit paru-paru dengan menyederhanakan tugas, mengurangi beban kerja ahli radiologi, dan meningkatkan akurasi deteksi nodul [2]. Teknik *machine learning* telah semakin banyak digunakan dalam bidang medis, khususnya dalam klasifikasi gambar untuk mempercepat deteksi penyakit dan mengurangi tenaga kerja manual dalam mendiagnosis penyakit pernapasan seperti kanker paru-paru [3]. Kemajuan ini bertujuan untuk meningkatkan deteksi penyakit secara dini, yang pada akhirnya dapat menyelamatkan nyawa dengan memungkinkan tindakan pencegahan dan intervensi pengobatan yang tepat waktu. Namun, kinerja dari model-model ini sangat tergantung pada proses optimasi yang digunakan selama pelatihan model. Pilihan pengoptimal secara signifikan berdampak pada hasil pelatihan model, yang mempengaruhi kecepatan konvergensi dan stabilitas [4][5]. *Optimizer* yang berbeda menunjukkan kinerja yang berbeda-beda berdasarkan karakteristik spesifik dari dataset dan kompleksitas model. Penelitian ini menekankan pentingnya pemilihan *optimizer* dalam *machine learning*, khususnya pada implementasi *deep learning*, di mana *optimizer* seperti Adam, BFGS, dan LM telah dibandingkan dari segi efektivitasnya [6].

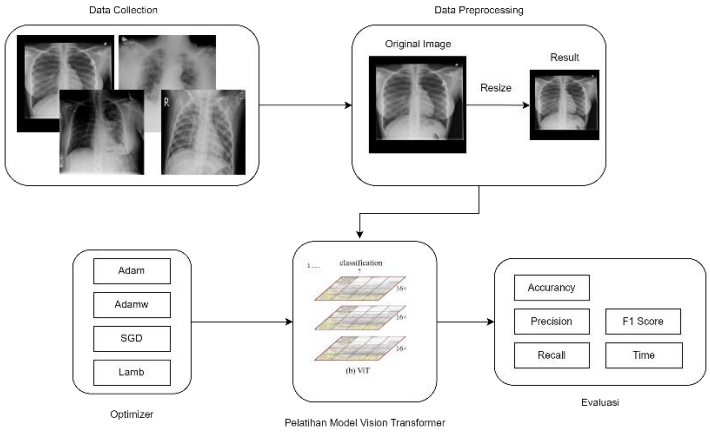
Masalah utama yang ditangani dalam penelitian ini adalah efektivitas dari *optimizer-optimizer* seperti Adam, AdamW, SGD, dan LAMB dalam konteks khusus klasifikasi penyakit paru-paru menggunakan model *vision transformer*. Penelitian ini bertujuan untuk mengevaluasi dan membandingkan kinerja masing-masing *optimizer* serta menentukan mana yang paling efisien dan akurat dalam konteks aplikasi medis ini.

Metode yang digunakan meliputi eksperimen komparatif di mana model *vision transformer* dilatih dengan setiap *optimizer* pada *dataset* yang seragam dari gambar rontgen dada, dengan variabel-variabel

seperti jumlah *epoch*, *batch size*, dan *learning rate* disetarakan untuk semua percobaan. Pengukuran kinerja akan fokus pada metrik-metrik seperti akurasi, *precision*, *recall*, dan *F1-score*.

2. Metode Penelitian

Metode penelitian ini dirancang untuk menganalisa proses klasifikasi penyakit paru-paru dengan menggunakan model *vision transformer* seperti pada Gambar 1.

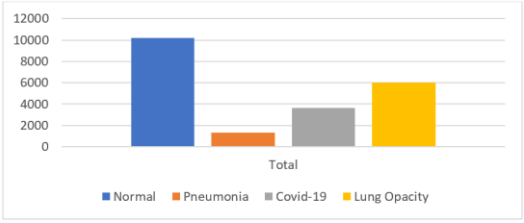


Gambar 1. Metode Penelitian

Penelitian ini menerapkan sebuah metodologi yang terstruktur untuk membandingkan efektivitas berbagai *optimizer* dalam klasifikasi penyakit paru-paru menggunakan citra X-ray dengan model *vision transformer*, dimulai dari *pra-pemrosesan* data hingga evaluasi kinerja model menggunakan metrik seperti akurasi, *precision*, *recall*, *F1-score* serta mempertimbangkan efisiensi waktu pelatihan.

2.1. Pengumpulan Dataset

Penelitian ini menggunakan dataset yang diambil dari *keagle*, yang dapat diakses melalui link https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database, dengan komposisi data merujuk Gambar 2.



Gambar 2. Komposisi Dataset

Dataset ini terdiri dari 21.165 citra x-ray dada, dengan pembagian sebagai berikut: 3.616 citra kasus COVID-19 positif, 10.192 citra normal, 6.012 citra Lung Opacity (bukan COVID-19), dan 1.345 citra pneumonia. [7],[8]

2.2. Preprocessing Data

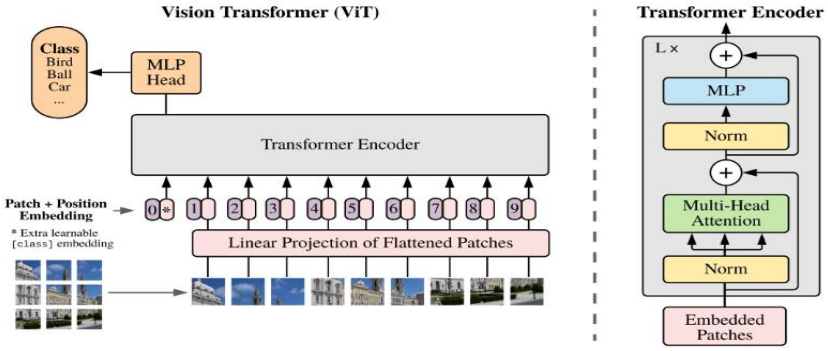
Dalam penelitian ini, kami menggunakan model *vision transformer* yang berbasis pada pendekatan yang dikembangkan oleh Alexey Dosovitskiy et al [9]. Semua gambar diubah ukurannya ke resolusi standar 224 x 224 piksel. Langkah normalisasi ini dimaksudkan untuk memastikan bahwa semua citra memiliki dimensi yang seragam, menghasilkan format yang homogen yang selaras dengan spesifikasi yang terverifikasi dalam penelitian sebelumnya tentang model *vision transformer*.

2.3. Pelatihan Model *Vision Transformer*

Vision Transformer (ViT) adalah model *deep learning* yang memproses gambar dengan membaginya menjadi beberapa bagian dengan ukuran tetap, dan memperlakukannya sebagai token untuk dianalisis [9]. Arsitektur *Vision Transformer* terdiri dari 3 komponen utama yaitu *patch + position*

embedding, *transformer encoder* dan *classification head*. Pada komponen pertama gambar input dibagi menjadi beberapa *patch*.

Merujuk pada Gambar 3, gambar dipotong menjadi bagian-bagian kecil yang sama ukurannya. Setiap patch diratakan menjadi vektor dan diberi proyeksi linier untuk mengubahnya menjadi bentuk yang sesuai untuk dimasukkan ke dalam *transformer encoder*. Setiap vektor patch ini kemudian diberi *positional encoding*, yang memberi tahu model di mana setiap *patch* berada di dalam gambar secara relatif. Sebuah token kelas tambahan dimasukkan dalam urutan. Ini bertindak sebagai *placeholder* untuk representasi global gambar yang akan digunakan untuk klasifikasi (*Extra Learnable [class] Embedding*).



Gambar 3. Arstitektur Model *Vision Transformer* [9]

Proses selanjutnya adalah ekstraksi fitur menggunakan bagian yang disebut dengan *encoder* dari *transformer*. *Encoder* ini terdiri dari beberapa blok, di mana setiap blok memiliki dua bagian utama. Bagian pertama adalah *Multi-Head Self-Attention*, yang memungkinkan model untuk melihat dan membandingkan semua *patch* gambar secara bersamaan. Fitur ini sangat berguna karena dapat membantu model memahami hubungan antar bagian gambar dan menemukan fitur penting yang mungkin tersebar di seluruh gambar.

Bagian kedua dalam blok tersebut adalah MLP (*Multi-Layer Perceptron*). MLP adalah jaringan saraf yang terdiri dari beberapa lapisan dan berfungsi untuk menambahkan kompleksitas pada proses model, sehingga informasi yang didapat dari *attention mechanism* dapat diolah menjadi representasi yang lebih mendalam. Setiap blok dalam *encoder* juga dilengkapi dengan lapisan normalisasi (*Norm*), yang terletak setelah *self-attention* dan MLP. Lapisan ini membantu menstabilkan pembelajaran model dengan menormalkan distribusi output.

Setelah semua *patch* gambar diproses oleh *encoder*, informasi yang dihasilkan kemudian diproses lebih lanjut oleh MLP *Head*. MLP *Head* ini terdiri dari satu atau lebih lapisan *dense neural network* yang menggunakan fungsi-fungsi *non-linear* untuk mengolah informasi tersebut menjadi sebuah representasi global gambar, berdasarkan *token* klasifikasi. Representasi global ini kemudian digunakan untuk membuat prediksi akhir, sebuah distribusi *probabilitas* yang menunjukkan kemungkinan gambar termasuk ke dalam kategori tertentu yang telah ditentukan selama pelatihan. Prediksi ini sangat penting karena menjadi dasar untuk keputusan klasifikasi akhir model, di mana model menentukan kategori mana yang paling mungkin sesuai dengan objek dalam gambar berdasarkan apa yang telah dipelajarinya sebelumnya.

2.4. *Optimizer*

Optimizer merupakan algoritme atau metode dalam kecerdasan buatan yang berperan penting dalam menyesuaikan parameter seperti bobot dan bias, dengan tujuan mengurangi fungsi kerugian atau meningkatkan efisiensi produksi, sehingga memfasilitasi perubahan nilai bobot dan penyesuaian laju pembelajaran dalam jaringan saraf agar kerugian dapat diminimalkan [10]. Oleh karena itu, pemilihan *optimizer* yang tepat dapat memberikan dampak signifikan terhadap akurasi dan efisiensi model.

2.4.1 Adam

Optimizer Adam adalah salah satu metode optimisasi yang sangat populer dalam pembelajaran mesin, terutama untuk melatih *deep neural network* berbagai aplikasi *deep learning*. Adam, singkatan dari "*Adaptive Moment Estimation*," secara efektif menggabungkan aspek-aspek terbaik dari dua optimasi sebelumnya yaitu RMSProp dan Momentum untuk mencapai konvergensi yang cepat dan efisien. [11] *Optimizer* adam menghitung estimasi pertama (*mean*) dan kedua (*uncentered variance*) dari *gradien*, dan menggunakan kedua estimasi tersebut untuk mengatur *learning rate* untuk setiap parameter. Rumusnya meliputi:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) . g_t \quad (1)$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) . g_t^2 \quad (2)$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (3)$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (4)$$
$$\theta_{t+1} = \theta_t - \frac{\alpha . \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (5)$$

Di mana m_t adalah estimasi momen pertama (mean) dari gradien, v_t adalah estimasi momen kedua (variansi tak terpusat) dari gradien, g_t adalah gradien pada waktu t, β_1 dan β_2 adalah koefisien peluruhan eksponensial untuk estimasi momen, θ adalah parameter yang akan dioptimalkan, α adalah *learning rate* dan ϵ adalah konstanta yang sangat kecil untuk mencegah pembagian dengan nol

2.4.2 Adamw

Optimizer AdamW adalah sebuah variasi dari optimizer Adam yang memperkenalkan teknik Weight Decay secara lebih eksplisit dalam proses *optimisasi* [12]. AdamW memodifikasi Adam dengan memisahkan *weight decay* dari proses update parameter. Ini mengatasi beberapa kekurangan Adam dalam mengelola *regularisasi weight decay*. Rumus AdamW serupa dengan Adam, tetapi *weight decay* diterapkan langsung ke parameter sebelum pembaruan berikutnya:

$$\theta_{t+1} = \theta_t - \frac{\alpha . \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \alpha\, \omega d\, \theta_t \quad (6)$$

Di mana ωd adalah *weight decay* yang diterapkan pada parameter.

2.4.3 Lamb

Optimizer LAMB (*Layer-wise Adaptive Moments optimizer for Batch training*) adalah teknik optimisasi yang dirancang khusus untuk meningkatkan pelatihan model-model *deep learning* berskala besar[13]. LAMB menggunakan beberapa konsep dari Adam, termasuk estimasi momentum dan skala gradien, namun dengan penyesuaian pada skala layer atau per-layer *normalization*, untuk menyesuaikan *learning rate* secara dinamis per layer. Rumus untuk update parameter pada LAMB adalah sebagai berikut:

$$r_t = \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon} \quad (7)$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\|r_t + \lambda\theta_t\|} (r_t + \lambda\theta_t) \quad (8)$$

Di mana λ adalah faktor *weight decay*, r_t adalah rasio perubahan yang diinginkan untuk setiap parameter, η adalah *learning rate* global, dan $\|.\|$menunjukkan norma *Euklidean* dari vektor.

3 SGD

Optimizer Stochastic Gradient Descent (SGD) adalah salah satu metode optimisasi dasar yang paling banyak digunakan dalam *machine learning*, khususnya untuk pelatihan model-model *deep learning* [14]. SGD adalah versi *stochastic* dari *gradient descent* yang melakukan pembaruan parameter menggunakan *subset* data acak. Rumus dasar SGD adalah sebagai berikut:

$$\theta_{t+1} = \theta_t - \eta\, g_t \quad (9)$$

Di mana θ_t adalah parameter yang akan dioptimalkan, η adalah *learning rate* dan g_t adalah gradien dari fungsi kerugian yang diestimasi pada subset data.

4 Hasil dan Pembahasan

Bab ini memaparkan hasil penelitian dan membahas implikasi serta rekomendasi berdasarkan analisis data yang telah dilakukan. Model *vision transformer* dievaluasi menggunakan lima *optimizer* berbeda: Adam, AdamW,, SGD, dan LAMB. Evaluasi dilakukan selama 50 epoch, batch size 32 dan *learning rate* 0.0001, dengan pencatatan metrik untuk akurasi, presisi, recall, skor F1, dan durasi epoch.

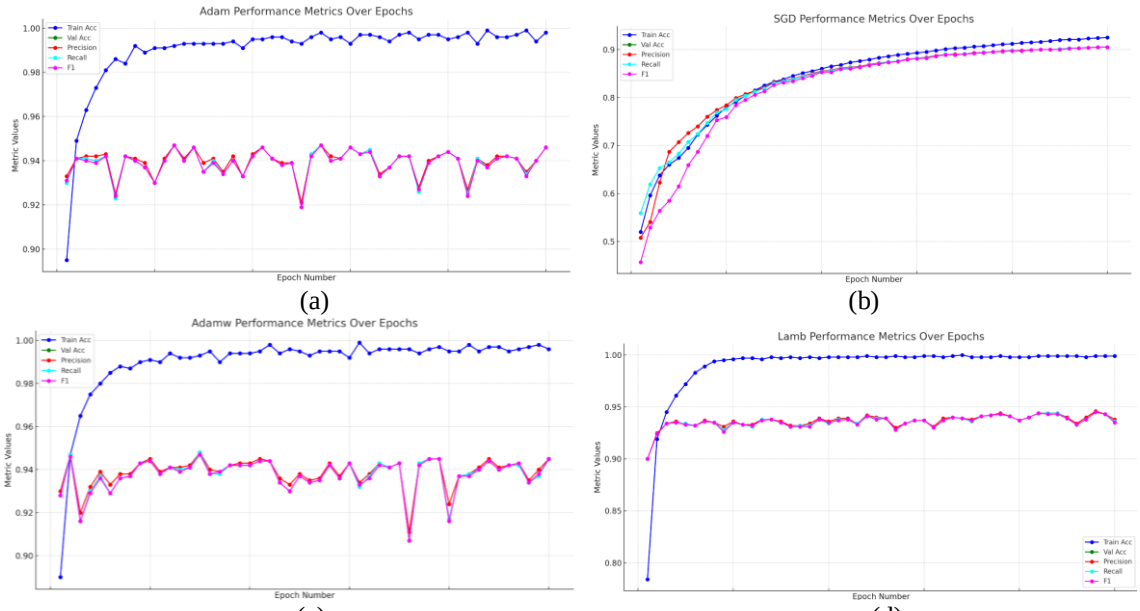
Tabel 1. Tabel rata-rata hasil ujicoba masing-masing *optimizer*

Optimizer	Training Acc	Validation Acc	Precision	Recall	F1-Score	Duration (sec)
Adam	0.990	0.939	0.940	0.939	0.939	652.71
AdamW	0.990	0.938	0.939	0.938	0.938	661.46
Lamb	0.989	0.936	0.937	0.936	0.936	689.94
SGD	0.843	0.836	0.836	0.823	0.823	670.28

Merujuk pada tabel 1 *optimizer* Adam dan AdamW juga menampilkan hasil yang mengesankan dengan akurasi validasi, presisi, dan F1-score yang hampir sama. Kemudian LAMB, meskipun memiliki akurasi pelatihan yang tinggi, sedikit tertinggal dalam akurasi validasi dan skor F1 dibandingkan dengan *optimizer* berbasis Adam. Sedangkan SGD, sebagai *optimizer* tradisional, menunjukkan performa yang paling rendah dalam semua metrik. Ini menegaskan bahwa optimasi lebih canggih seperti yang ditemukan dalam varian Adam dapat lebih efektif untuk jaringan yang kompleks seperti *vision transformer*.

Selain itu terkait *duration* terlihat bahwa LAMB membutuhkan waktu terpanjang dalam melaksanakan satu *epoch*, yaitu rata-rata 689.94 detik, sedangkan Adam tercatat sebagai yang tercepat dengan rata-rata waktu 652.71 detik per epoch. *Optimizer* AdamW dan SGD menunjukkan performa waktu yang serupa, berada di antara kinerja Adam dan LAMB. Kecepatan Adam menandakan kemungkinan efisiensi komputasional yang lebih tinggi dibandingkan dengan *optimizer* lain, walaupun perbedaan waktunya tidak terlampau signifikan, terkecuali untuk LAMB yang secara konsisten memakan waktu lebih lama.

Pada Gambar 4 grafik seluruh *optimizer* menunjukkan peningkatan performa seiring dengan berjalannya epoch, meskipun dengan tingkat fluktuasi yang berbeda. lamb dan SGD menunjukkan tren yang sangat stabil yang menandakan konvergensi yang efisien dan konsisten. Sedangkan Adam dan AdamW menampilkan fluktuasi yang lumayan tinggi.



(a) (b)

(c) (d)

Gambar 4. Grafik Trend Optimizer Adam (a), SGD (b), Adamw(c), Lamb(d)

5 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan, optimizer memainkan peran kritis dalam efektivitas pelatihan model *Vision Transformer* untuk klasifikasi penyakit paru-paru. Dalam analisis ini, empat *optimizer* yaitu Adam, AdamW, SGD, dan LAMB dibandingkan untuk menilai performa mereka dalam konteks kecepatan dan efisiensi pelatihan. *Optimizer* Adam terbukti memiliki kecepatan pelatihan yang paling cepat dengan rata-rata 652.71 detik per *epoch*, menunjukkan efisiensi komputasi yang paling baik. Sebaliknya, LAMB menunjukkan waktu pelatihan paling lama, dengan rata-rata 689.94 detik per *epoch*, yang bisa mengindikasikan beban komputasi yang lebih besar. SGD, yang waktu pelatihannya serupa dengan LAMB, menawarkan performa yang kurang optimal namun tetap relevan sebagai alternatif ketika stabilitas lebih diutamakan daripada efisiensi. AdamW, yang merupakan modifikasi dari Adam, juga menunjukkan hasil yang baik, menegaskan bahwa penyesuaian parameter walaupun kecil dapat memiliki dampak yang signifikan terhadap hasil pelatihan model. Oleh karena itu, pemilihan *optimizer* yang tepat sangat penting dan harus disesuaikan dengan kebutuhan khusus terkait dengan kecepatan dan stabilitas dalam aplikasi nyata..

Daftar Pustaka

[1] K. A, G. N. R, D. D, and K. A, "Analysis and Classification of the Lung Cancer with CNN Implementation," in *2022 Smart Technologies, Communication and Robotics (STCR)*, 2022, pp. 1–4. doi: 10.1109/STCR55312.2022.10009558.

[2] O. Khouadja and M. S. Naceur, "Lung Cancer Detection with Machine Learning and Deep Learning: A Narrative Review," in *2023 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, 2023, pp. 1–8. doi: 10.1109/IC_ASET58101.2023.10150913.

[3] S. K. H. Bukhari and L. Fahad, "Lung Disease Detection using Deep Learning," in *2022 17th International Conference on Emerging Technologies (ICET)*, 2022, pp. 154–159. doi: 10.1109/ICET56601.2022.10004651.

[4] N. Fonseca, V. Guidetti, and W. Trojak, "Probing optimisation in physics-informed neural networks," pp. 1–16, 2023.

[5] S. Bashetty, K. Raja, S. Adepu, and A. Jain, "Optimizers in Deep Learning: A Comparative Study and Analysis," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 12, pp. 1032–1039, 2022, doi: 10.22214/ijraset.2022.48050.

[6] J. Taylor, W. Wang, B. Bala, and T. Bednarz, "Optimizing the optimizer for data driven deep neural networks and physics informed neural networks," 2022, [Online]. Available: http://arxiv.org/abs/2205.07430

[7] T. Rahman *et al.*, "Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images," *Comput. Biol. Med.*, vol. 132, no. March, p. 104319, 2021, doi: 10.1016/j.compbiomed.2021.104319.

[8] M. E. H. Chowdhury *et al.*, "Can AI Help in Screening Viral and COVID-19 Pneumonia?," *IEEE Access*, vol. 8, pp. 132665–132676, 2020, doi: 10.1109/ACCESS.2020.3010287.

[9] A. Dosovitskiy *et al.*, "an Image Is Worth 16X16 Words: Transformers for Image Recognition At Scale," *ICLR 2021 - 9th Int. Conf. Learn. Represent.*, 2021.

[10] P. D. Wulaning Ayu and G. A. Pradipta, "U-Net Tuning Hyperparameter for Segmentation in Amniotic Fluid Ultrasonography Image," *2022 4th Int. Conf. Cybern. Intell. Syst. ICORIS 2022*, no. June, 2022, doi: 10.1109/ICORIS56080.2022.10031294.

[11] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[12] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *7th Int. Conf. Learn. Represent. ICLR 2019*, 2019.

[13] Y. You *et al.*, "Large Batch Optimization for Deep Learning: Training Bert in 76 Minutes," *8th Int. Conf. Learn. Represent. ICLR 2020*, 2020.

[14] L. Bottou, "Stochastic gradient descent tricks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7700 LECTU, no. 1, pp. 421–436, 2012, doi: 10.1007/978-3-642-35289-8_25.