

# Fire Detection Using Learning-Without-Forgetting

Steven Adrian Gracia<sup>1</sup>

<sup>1</sup>*Informatics, Petra Christian University*

<sup>1</sup>c14210171@john.petra.ac.id

**Abstract**— Kebakaran hutan berdampak buruk kepada area perhutanan sehingga harus ditangani secepat mungkin. Salah satu cara untuk mendeteksi kebakaran tersebut adalah menggunakan deep learning. Eksperimen yang dilakukan dalam laporan ini menggunakan model EfficientNetV2S untuk mendeteksi kebakaran. Dataset yang digunakan berupa Forest Fire and Non Fire serta BoWFire yang merupakan dataset yang menantang. Dengan metode fine tuning, model dapat meraih akurasi 85.02% di dataset Forest Fire dan 63.72% di dataset BoWFire. Training model untuk mengenali gambar-gambar yang lebih sulit dalam BoWFire dibedakan dilakukan dengan beberapa cara. Pelanjutan training memberikan hasil 72.93% untuk data lama dan 78.76% untuk BoWFire. Hal ini terjadi karena metode ini cenderung overfitting. Mengulang training dengan dataset campuran memberikan hasil 82.47% dan 62.39%. Joint training memberikan hasil 51.00% dan 95.58% namun hasil tersebut merupakan hasil eksperimen yang memiliki kendala memori akibat konversi dataset ke array. Learning without forgetting digunakan untuk melatih model dengan data baru tanpa melupakan data lama. Dengan metode ini, model dapat meraih akurasi 81.74% di dataset Forest Fire dan 80.53% di dataset BoWFire.

**Keywords**— Fire Detection, Deep Learning, Learning-Without-Forgetting, Image Processing

## I. INTRODUCTION

Kebakaran hutan adalah salah satu penyebab aktif berkurangnya luas hutan di dunia. Ditambah dengan pemanasan global dan penggundulan, area perhutanan sudah berkurang menjadi setengah dari areanya pada satu abad yang lalu [1]. Kebakaran hutan berdampak pada kerusakan habitat tanaman dan hewan serta penurunan kualitas tanah. Untuk mencegah hal ini, banyak upaya yang sudah dilakukan termasuk mendeteksi, menganalisa, dan mensimulasikan kebakaran tersebut. Dengan perkembangan teknologi, deep learning menjadi salah satu sarana yang dapat membantu upaya tersebut.

Salah satu metode yang digunakan untuk mendeteksi kebakaran hutan oleh [2] adalah neural architecture search-based object detection (DetNAS) untuk mencari backbone yang optimal serta model BNN untuk memperkirakan area yang rusak. Hasil yang didapatkan adalah penggunaan Faster R-CNN dengan mean average precision 27.9. Dalam penelitian lain, [3] menggunakan transfer learning dan learning without forgetting untuk mengubah model pre-trained supaya dapat berfokus kepada deteksi kebakaran. Model terbaik dalam penelitian ini yaitu Xception berhasil mendapatkan akurasi 98.72% dengan dataset umum dan akurasi 91.41% dengan dataset sulit dengan akurasi

96.89% di dataset awal setelah dilakukan LwF. Selain itu, [4] menggunakan InceptionResNetv2 untuk mencapai akurasi 99.07% untuk dataset api dan asap.

Selain menggunakan gambar nyata, [5] juga menggunakan data eksperimen api untuk menentukan heat release rate (HRR) dari sebuah fenomena kebakaran. Informasi ini digunakan untuk mengidentifikasi tahap penyebaran api secara real-time. Model yang sudah ditrain dapat memprediksi HRR dengan tepat, yaitu dengan minimum MSE loss 885. Penelitian lain dalam [6] menggunakan arsitektur SmokeyNet untuk menggabungkan informasi dari beberapa frame dalam sebuah video dan membaginya menjadi beberapa bagian. Akurasi yang didapatkan dengan backbone ResNet34 dan input 2 frame sebesar 83.49%, dengan peningkatan akurasi jika digunakan backbone yang lebih besar atau jumlah frame yang lebih banyak yang disertai pertambahan waktu inferensi.

Dalam proyek ini, saya mengusulkan pembuatan sistem deteksi kebakaran hutan menggunakan model deep learning yaitu EfficientNet[7]. Percobaan dilakukan dengan feature extraction dan fine tuning untuk memanfaatkan model pre-training, lalu menggunakan learning without forgetting untuk mencoba menangani dataset yang lebih sulit. Hal ini dilakukan untuk melatih model yang lama dengan data yang baru dengan menghindari hilangnya kemampuan model untuk mengenali data lama. Kinerja dari beberapa metode juga akan dibandingkan untuk menemukan metode yang paling cocok.

## II. DATASET

Ada 2 dataset yang digunakan dalam proyek ini. Forest Fire and Non Fire Dataset[8] merupakan dataset besar yang digunakan untuk mentrain model api umum. Dataset ini berisi total 21.7 ribu foto yang dibagi menjadi ‘fire’ dan ‘non fire’. 7805 dari masing-masing kelas merupakan data untuk training, sedangkan 2666 gambar ‘fire’ dan 3457 gambar ‘non fire’ tersedia untuk testing. Contoh dari dataset ini dapat dilihat di Fig. 1



Fig. 1 Contoh ‘fire’ dan ‘non fire’ dari dataset [8]

BoWFire[9] adalah dataset kecil yang hanya berisi 226 gambar saja yang dibagi menjadi 119 gambar ‘fire’ dan 107 gambar ‘not\_fire’. Dataset ini termasuk dataset yang sulit karena banyak dari gambar non-api yang memiliki ciri-ciri api seperti warna jingga terang dan sumber cahaya yang dominan. Contoh dari dataset ini dapat dilihat di Fig. 2



Fig. 2 Contoh ‘fire’ dan ‘not\_fire’ dari dataset [9]

Untuk kedua dataset tersebut, preprocessing dilakukan dengan memutar, menggeser, memperbesar/memperkecil, dan melakukan penskalaan warna RGB pada setiap gambar yang ada. Preprocessing dilakukan dengan ImageDataGenerator dari Keras yang melakukan preprocessing secara real time saat training. ImageDataGenerator dapat meringankan beban memori, terutama dengan dataset besar seperti Forest Fire and Non Fire Dataset yang berisi 3.75 GB.

### III. PROPOSED METHOD

Percobaan ini hanya menggunakan satu jenis model yaitu EfficientNet, khususnya EfficientNetV2S[10][11]. Model ini dipilih karena ukurannya yang cukup kecil serta waktu training yang relatif cepat dengan hasil yang lebih akurat dari beberapa model lainnya. Percobaan dalam proyek ini berfokus kepada metode training yang dilakukan, serta konfigurasi untuk setiap metode tersebut.

#### A. Transfer Learning

Transfer learning adalah suatu metode machine learning dimana kemampuan dari model yang sudah ditrain diaplikasikan ke model baru[12]. Metode ini dilakukan dengan menginisialisasi model dengan weight yang didapatkan dari hasil training lain lalu menggunakan sebagai feature extractor atau melakukan fine tuning kepada beberapa layer akhir. Dengan cara ini, model dapat memiliki kemampuan yang cukup baik tanpa training dari awal sehingga menghemat sumber daya komputasi.

#### B. Feature Extraction

Feature extraction adalah penggunaan model pre-trained untuk mengekstrak fitur penting dari sebuah data. Dengan metode ini, hanya layer klasifikasi akhir yang perlu diganti dan dilatih ulang untuk menyesuaikan dengan data yang baru.

#### C. Fine Tuning

Metode ini mirip seperti feature extraction, namun beberapa layer akhir dari model ditrain ulang. Dengan cara ini, model

dapat mengenali fitur yang lebih relevan ke data yang akan diproses. Layer awal tidak perlu ditrain ulang karena fitur-fitur yang dikenali cenderung umum sehingga training tidak memakan waktu yang terlalu lama.

#### D. Joint Training

Dalam joint training, sebuah model dilatih untuk melakukan beberapa task secara bersamaan. Hasil output dari model tersebut dapat digabung untuk menentukan prediksi akhir. Joint learning dapat menggunakan data yang berbeda dan memberikan output terpisah untuk setiap tugas yang dipelajari dalam suatu masalah. Joint training bisa menjadi cara yang paling akurat, namun training yang harus dilakukan menguras banyak sumber daya komputasi.

#### E. Learning Without Forgetting

Transfer learning cenderung melakukan overfitting pada data baru karena parameter yang sudah ditrain dengan data lama dilupakan dan diubah. Hal ini dapat menyebabkan hilangnya kemampuan untuk melakukan tugas lama dengan tepat setelah training. Learning without forgetting (LwF) berupaya untuk mentrain model dengan data baru tanpa melupakan kemampuan lama untuk mencari akurasi terbaik diantara dua tugas tersebut. Salah satu keunggulan dari metode ini adalah tidak adanya kebutuhan untuk data lama. Proses LwF adalah sebagai berikut untuk model yang memiliki parameter lama  $\theta_o$  dan parameter baru  $\theta_n$  dengan data baru  $X_n$  dan label  $Y_n$ :

- 1) Cari output  $Y_o$  dari model lama untuk data baru.
- 2) Inisialisasi  $\theta_n$  dengan weight acak, pastikan untuk menambahkan node baru untuk setiap label baru jika ada.
- 3) Train model dengan menggunakan  $Y_o$  dan  $Y_n$  untuk menghitung loss sehingga model tidak melupakan kemampuan lama.

### IV. EKSPERIMENT

Eksperimen dilakukan untuk meneliti performa model dengan metode feature extraction, fine tuning, joint training, dan learning without forgetting. Training dilakukan dengan runtime T4 yang dimiliki oleh Google Colab serta framework Tensorflow dan Keras. Seluruh model di train selama 20 epoch jika tidak terjadi early stopping, kecuali untuk metode LwF yang diuji selama 200 epoch. Early stopping terjadi kepada seluruh model yang tidak mengalami perkembangan akurasi atau loss selama beberapa epoch.

Model EfficientNetV2S yang digunakan dalam eksperimen ini menggunakan weight pre-trained dari ImageNet, diambil dari tensorflow.keras.applications. Layer klasifikasi dari model dihapus dan diganti dengan layer flatten, dropout, dense, dan output berupa sigmoid. Model juga dicompile menggunakan optimizer Adam dan perhitungan lossnya memakai binary cross entropy.

Layer yang ditrain pada metode feature extraction merupakan layer klasifikasi pengganti tersebut. Di sisi lain, fine tuning juga melatih ulang sebagian dari layer EfficientNetV2S. Feature extraction berhasil meraih akurasi

79.48%. Fine tuning memberikan hasil yang lebih baik yaitu akurasi 85.02%. Untuk metode-metode selanjutnya, weight dari fine tuning akan digunakan untuk melewati tahap training api awal. Perlu diketahui bahwa saat diuji dengan data BoWFire, model hasil fine tuning hanya dapat meraih akurasi 63.72% saja.

Hal pertama yang dicoba adalah melanjutkan training seluruh model dengan data BoWFire. Model ini berhasil meraih akurasi yang lebih tinggi di BoWFire dengan 78.76%, namun kemampuan model untuk menganalisa api biasa berkurang menjadi 72.93%. Hal ini terjadi karena seiring berjalannya training, model melupakan informasi lama yang berkaitan dengan dataset umum, dan berfokus kepada informasi yang spesifik kepada dataset BoWFire. Faktor lupa ini juga diperparah dengan ukuran dataset yang sangat kecil, sehingga overfitting dengan data yang baru sangat mudah terjadi.

Hal lain yang dicoba adalah memulai ulang training menggunakan dataset campuran antara data lama dan BoWFire. Konfigurasi model sama dengan yang dipakai di dalam metode fine tuning. Kelemahan dari metode ini adalah perlunya dataset awal dan sumber daya untuk melakukan training dari awal, meskipun sudah ada model pre-trained hasil fine tuning. Hasil yang didapatkan dari metode ini adalah akurasi 82.47% dengan data lama dan 62.39% dengan BoWFire, sedikit lebih rendah dari fine tuning biasa. Hal ini mungkin terjadi karena variasi yang lebih luas dalam data training sehingga model kesulitan untuk mengerti fitur api dengan benar.

Metode selanjutnya adalah menggunakan joint training. Model joint terdiri dari model lama, model BoW, dan output layer. Model lama merupakan model yang sama persis dengan model fine-tuned tanpa dense dan output layer di akhir. Model BoW merupakan model dengan arsitektur yang sama dengan model lama namun ditrain ulang menggunakan dataset BoWFire. Layer dense dan output dari model ini juga dilepas. Hasil output dari kedua model tersebut diconcatenate di model keseluruhan, lalu dimasukkan di output layer bersama. Gambar arsitektur model tersebut dapat dilihat di Fig. 3.

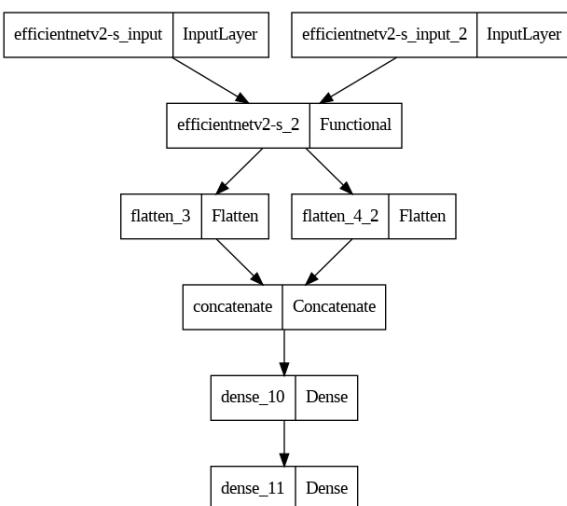


Fig. 3 Arsitektur model joint

Dalam proses training model, ditemukan bahwa metode ini tidak dapat menggunakan ImageDataGenerator karena input yang diterima hanya dalam bentuk numpy. Saat mengonversi dataset Forest Fire and Non Fire, runtime selalu mengalami crash. hal ini terjadi karena untuk membuat sebuah numpy array yang menyimpan 21 ribu foto dengan dimensi 256 x 256, 3 warna, dan 4 byte per value (float32) diperlukan memori sekitar 16.5 GB. Hal ini menunjukkan kelemahan dari joint training yaitu ketergantungannya kepada dataset lama. Tentunya, ketergantungan ini juga menyebabkan pemakaian sumber daya waktu dan penyimpanan yang boros, sama seperti training menggunakan dataset campur. Akhirnya, model hanya ditrain menggunakan dataset BoWFire dan 300 gambar dari dataset awal. Akibatnya, model memiliki akurasi yang tidak imbang yaitu 51.00% untuk dataset awal dan 95.58% untuk dataset BoWFire.

Metode terakhir adalah learning without forgetting. Karena metode ini hanya menggunakan data baru, sumber daya yang dipakai tidak terlalu banyak. Model yang digunakan ada 2 macam: old dan new. Kedua model menggunakan arsitektur dan weight dari hasil fine tuning di awal. Model old berfungsi sebagai model lama yang menjaga kemampuan model baru untuk tidak lupa. Model new adalah model yang diberikan data BoWFire sebagai training. Untuk melakukan learning without forgetting, perlu dibuat fungsi training custom. Loss yang dipakai adalah gabungan loss dari perbandingan prediksi dengan output model lama dan label sebenarnya. Rasio perhitungan loss 50:50 memberikan hasil terbaik dimana model new dapat memberikan akurasi 81.74% untuk data lama dan 80.53% untuk BoWFire, yang membuktikan bahwa model tersebut mempelajari data baru tanpa melupakan data lama. Dapat diperhatikan juga bahwa kenaikan drastis akurasi untuk dataset baru diikuti dengan penurunan akurasi untuk dataset lama yang relatif sedikit.

Perbandingan performa dari seluruh model dapat dilihat dalam Table 1.

TABLE I  
PERFORMA BERBAGAI METODE DENGAN DATASET BoWFire

Metode	Akurasi(data lama)	Akurasi(BoWFire)
Feature extraction	79.48%	-
Fine tuning	85.02%	63.72%
Continue training	72.93%	78.76%
Dataset Campur	82.47%	62.39%
Joint training*	51.00%	95.58%
Learning without forgetting	81.74%	80.53%

## V. KESIMPULAN

Untuk mendeteksi kebakaran, dapat dilakukan pendekatan berbasis deep learning. Model EfficientNetV2S merupakan salah satu model yang dapat dilatih untuk mengenali api dengan baik. Metode learning without forgetting dapat

memberikan sebuah model kemampuan untuk beradaptasi dengan data baru tanpa melupakan kemampuan sebelumnya. Selain peningkatan performa, metode ini juga tidak memerlukan data training lama sehingga sumber daya yang dilakukan untuk training lebih sedikit.

#### REFERENCES

- [1] F. Carta, Chiara Zidda, M. Putzu, D Loru, Matteo Anedda, and D. D. Giusto, "Advancements in Forest Fire Prevention: A Comprehensive Survey," *Sensors*, vol. 23, no. 14, pp. 6635–6635, Jul. 2023, doi: <https://doi.org/10.3390/s23146635>.
- [2] D. Q. Tran, M. Park, Y. Jeon, J. Bak, and S. Park, "Forest-Fire Response System Using Deep-Learning-Based Approaches With CCTV Images and Weather Data," *IEEE Access*, vol. 10, pp. 66061–66071, 2022, doi: <https://doi.org/10.1109/ACCESS.2022.3184707>.
- [3] V. E. Sathishkumar, J. Cho, M. Subramanian, and O. S. Naren, "Forest fire and smoke detection using deep learning-based learning without forgetting," *Fire Ecology*, vol. 19, no. 1, Feb. 2023, doi: <https://doi.org/10.1186/s42408-022-00165-0>.
- [4] Raghad K. Mohammed. "A real-time forest fire and smoke detection system using deep learning". *International Journal of Nonlinear Analysis and Applications*, vol. 13, no. 1, Mar. 2022, doi: <https://doi.org/10.22075/ijnaa.2022.5899>.
- [5] Z. Wang, T. Zhang, and X. Huang, "Predicting real-time fire heat release rate by flame images and deep learning," *Proceedings of the Combustion Institute*, Aug. 2022, doi: <https://doi.org/10.1016/j.proci.2022.07.062>.
- [6] Anshuman Dewangan et al., "FIgLib & SmokeyNet: Dataset and Deep Learning Model for Real-Time Wildland Fire Smoke Detection," *Remote Sensing*, vol. 14, no. 4, pp. 1007–1007, Feb. 2022, doi: <https://doi.org/10.3390/rs14041007>.
- [7] B. Koonce, "EfficientNet," Convolutional Neural Networks with Swift for Tensorflow, pp. 109–123, 2021, doi: [https://doi.org/10.1007/978-1-4842-6168-2\\_10](https://doi.org/10.1007/978-1-4842-6168-2_10).
- [8] Amerzish Minha, "Forest Fire and Non Fire Dataset," Kaggle.com, 2023. <https://www.kaggle.com/datasets/amerzishminha/forest-fire-and-non-fire-dataset> (accessed Jun. 23, 2024).
- [9] "BoWFire Object Detection Dataset by Gachon," Roboflow. <https://universe.roboflow.com/gachon-naj3/bowfire-0fqng> (accessed Jun. 23, 2024).
- [10] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," arXiv:2104.00298 [cs], Jun. 2021, Available: <https://arxiv.org/abs/2104.00298v3>
- [11] "tf.keras.applications.EfficientNetV2S | TensorFlow v2.16.1," TensorFlow. [https://www.tensorflow.org/api\\_docs/python/tf/keras/applications/EfficientNetV2S](https://www.tensorflow.org/api_docs/python/tf/keras/applications/EfficientNetV2S) (accessed Jun. 23, 2024).
- [12] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, May 2016, doi: <https://doi.org/10.1186/s40537-016-0043-6>.