Fachbereich Informatik,
Kommunikation und Wirtschaft
Studiengang Wirtschaftsmathematik
HTW Berlin

From Scratch vs. PreTrained: Deep Learning Approaches to MRI Brain Tumor Classification

Ausarbeitung zu Data Science
in Finance and Insurance

von
Phillip Olshausen

vorgelegt bei
Prof. Sunna Torge

Berlin, 3. Februar 2026

Inhaltsverzeichnis

1 Introduction

Brain tumors represent one of the most critical and complex challenges in medical diagnosis. The diverse nature, complex structure, and aggressive behavior of brain tumors significantly complicate accurate diagnosis and effective treatment planning. Traditional methods involving manual analysis of magnetic resonance imaging (MRI) scans by radiologists can be labor-intensive, subject to human error, and often with inconsistent results. The need for automated, efficient, and highly accurate diagnostic tools has become increasingly evident. The rapid advancements in artificial intelligence, particularly Convolutional Neural Networks (CNN), have revolutionized medical imaging and diagnostic practices. CNNs are known for their exceptional capabilities in image recognition and feature extraction and offer a powerful method for automatically identifying complex patterns in medical images. That makes them particularly well-suited for the classification of brain tumor types. As a result, they often outperform traditional manual methods in both efficiency and diagnostic reliability, especially with MRI scans. Several studies have demonstrated the effectiveness of CNN-based methodologies in tumor classification. Reddy and Dhuli (2023) illustrated that even a minimalist CNN structure could achieve impressive performance with MRI images, reporting 99.58% high accuracy rates despite the simplicity of their architecture. Their findings provided foundational insight into the capabilities of compact CNN structures for medical image classification tasks. Additionally, Khan and Auvee (2024) conducted a comparative analysis of a custom CNN against pretrained ResNet-18 and VGG-16, finding that the lightweight model attained up to 99.62% accuracy while significantly reducing computational requirements. Our research specifically targets the application of CNNs to the classification of brain tumors into distinct types, namely glioma, meningioma, pituitary tumors, and non-tumor images. We chose to explore two complementary approaches: firstly, developing a straightforward CNN model trained from scratch, and secondly, a pretrained CNN architecture, specifically EfficientNet-B0. We will evaluate both models on our labeled MRI dataset, compare their classification accuracy and training efficiency, and assess whether transfer learning with EfficientNet-B0 offers a significant performance advantage over the scratch-built CNN.

2 Convolutional Neural Network

2.1 Theoretical Foundations

Convolutional Neural Networks (CNNs) are specialized deep learning models for image analysis, leveraging spatial hierarchies to learn visual features. A convolutional layer is the core building block in CNNs, extracting spatial patterns by applying a learnable kernel over every position in the input feature map (Reddy and Dhuli, 2023) . If B denotes the input map and C the filter kernel, then the output feature map F at coordinates (u, v) is given by

$$F(u, v) = (B * C)(u, v) = \sum_{m} \sum_{n} B(u, v)\, C(u - m,\, v - n).$$

Here $*$ denotes the convolution operator, and (u, v) indexes the spatial location in the resulting map.

Each convolutional block in our network is immediately followed by a 2×2 max-pooling operation. Max pooling partitions the feature map into non-overlapping 2×2 regions and retains only the maximum activation from each region. This halves the spatial dimensions of the map, which makes the network less sensitive to small translations, reduces the number of parameters, and helps prevent overfitting by forcing the network to focus on the strongest responses.

After pooling, we introduce non-linearity with the Rectified Linear Unit (ReLU) activation, defined by

$$ReLU(x) = max(0, x).$$

ReLU not only enables the network to learn more complex functions but also helps avoid the vanishing gradient problem, allowing gradients to propagate effectively through many layers. Furthermore, a softmax function converts the final set of scores $\mathbf{z} = [z_1, z_2, z_3, z_4]$ into a probability distribution p over the four classes of tumors. The network is trained by minimizing the categorical cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{4} y_i \, \log p_i,$$

where $\mathbf{y}$ is the one-hot encoded true label. Training is performed using the Adam optimizer, an adaptive stochastic gradient algorithm that maintains per-parameter learning rates and momentum to accelerate convergence and improve robustness.To guard against overfitting without increasing complexity, we place a dropout layer (rate = 0.5) immediately after the 128-unit

dense layer. At each training step, dropout randomly deactivates half of the neurons in that layer, which discourages the model from relying too heavily on any one feature. Overall, our architecture consists of three convolutional blocks with 32, 64 and 128 filters respectively, each using a 3×3 kernel each followed by 2×2 max-pooling. The flattened output then feeds into a fully connected layer with 128 ReLU units, a dropout (rate = 0.5) and the final dense layer with four units and softmax activation. The CNN architecture implemented in this study is defined in Algorithm 1 as follows:

Algorithm 1: CNN Model

```
def build_model(img_size=IMG_SIZE, n_classes=len(CLASSES)):
     model = models.Sequential([
         layers.Input(shape=(img_size[0], img_size[1], 1)),
         layers.Conv2D(32, kernel_size=3, activation="relu"),
         layers.MaxPooling2D(pool_size=2),
         layers.Conv2D(64, kernel_size=3, activation="relu"),
         layers.MaxPooling2D(pool_size=2),
         layers.Conv2D(128, kernel_size=3, activation="relu"),
         layers.MaxPooling2D(pool_size=2),
         layers.Flatten(),
         layers.Dense(128, activation="relu"),
         layers.Dropout(0.5),
         layers.Dense(n_classes, activation="softmax")
     ])
     model.compile(optimizer="adam",
     loss="categorical_crossentropy",
     metrics=["accuracy"])
     return model
```

2.2 Dataset and Preprocessing

The images used in this study were obtained from a publicly available dataset curated by Ghaffar and Ayesha (2024) on Mendeley Data, titled "Brain Tumor Data". In total, the dataset comprises 7023 images split into training and testing subsets. The training set includes 5712 scans, of which 1321 (23.1 %) depict gliomas, 1339 (23.4%) meningiomas, 1595 (27.9 %) normal (no-tumor)

brains, and 1457 (25.5%) pituitary tumors. The testing set consists of 1311 scans, with 300 glioma images (22.9%), 306 meningioma images (23.3 %), 405 normal images (30.9%), and 300 pituitary tumor images (22.9%). This near-balanced class distribution ensures that no single category dominates, thus reducing bias during training.

Prior to model ingestion, all scans were uniformly resized to 150×150 pixels to standardize spatial dimensions and reduce computational load. Each image was then converted to grayscale in order to emphasize textural and structural information over color variations, which are less informative in MRI scans. Min–max normalization was applied per image to scale pixel values to the [0, 1] range, mitigating intensity discrepancies across scans and enabling the network to focus on meaningful structural differences rather than scanner-specific brightness variations.

2.3 Training and Validation Strategy

The development of our CNN architecture began with the use of five-fold stratified cross-validation. This method partitions the training set into five mutually exclusive subsets, preserving the original class proportions in each fold. In each iteration, four folds were used to train the network over a fixed number of epochs, while the remaining fold served as a temporary validation set to track performance. Recording the peak validation accuracy for each fold and averaging across folds yielded a mean cross-validation accuracy of 92.14% with a standard deviation of 1.58%, indicating stable performance and low sensitivity to specific train–validation splits. We avoided a static train–validation split at this stage to prevent biased performance estimates that can result from chance imbalances in class representation.

After finishing our cross-validation tests, we trained on the full dataset while holding back a small set of samples to keep an eye on performance. Specifically, 10% of the training data was randomly separated using a stratified split to form a holdout "monitor" set, with the remaining 90% used to update model weights. Training proceeded for ten epochs with a batch size of 32 and the Adam optimizer, during which the monitor set provided real-time feedback on validation loss and accuracy. This strategy enabled us to detect emerging signs of overfitting. Our chosen threshold of a 5% gap between training and validation accuracy was never exceeded and no overfitting warnings were triggered.

Algorithm 2: Final Training with Monitor Split

```
y_int = np.argmax(y_train, axis=1)
X_tr, X_mon, y_tr, y_mon = train_test_split(
    X_train, y_train, test_size=0.10, stratify=y_int, random_state=42
)
EPOCHS_FINAL = 10
final_model = build_model()
final_model.summary()
hist_final = final_model.fit(
    X_tr, y_tr,
    epochs=EPOCHS_FINAL, batch_size=BATCH,
    validation_data=(X_mon, y_mon),
    shuffle=True, verbose=1
)
```

By adopting the monitor-split approach rather than a fixed cross-validation or a one-off validation split, we were able to combine the advantages of model robustness assessment with practical, single-pass training on all available data. This approach both confirmed our model's stability and made full use of the available data, as shown by the strong learning curves and high test accuracy reported in the Initial Results and Missclassifications section.

2.4 Initial Results and Missclassifications

The final model was evaluated on the held-out test set from the monitor split and achieved an accuracy of 96.34%. This result confirms the strong performance seen during validation. Such high test accuracy demonstrates that the model successfully transferred its learned representations to unseen data, indicating robustness against variations in the dataset.

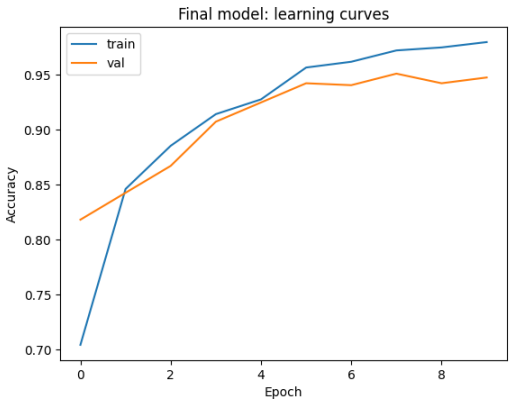

Abbildung 1: Training and Validation Accuracy (CNN)

Figure 1 illustrates the training and validation accuracy over the ten epochs, highlighting the model's learning dynamics. The training history of the final CNN model shows a consistent rise in both training and validation accuracy over ten epochs. In the first epoch, training accuracy began at 70.43%, while validation accuracy stood at 81.82%. By the fifth epoch, training accuracy had climbed to 92.76%, with validation accuracy reaching 92.48%. The highest validation accuracy 95.10% was observed in epoch eight, at which point training accuracy was 97.22%. At epoch ten, the model converged to a training accuracy of 97.98% and a validation accuracy of 94.76%. The gap between final training and validation accuracy, which was approximately 3.22 percentage points, remained below our 5% overfitting threshold. No overfitting warnings were issued, indicating that the model retained its ability to generalize throughout training.

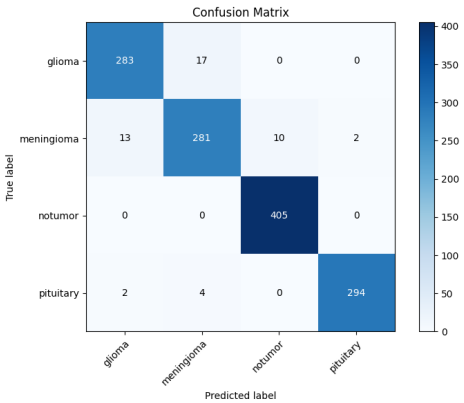

Abbildung 2: CNN Confusion Matrix

Figure 2 presents the confusion matrix computed on the held-out test set, revealing the distribution of true versus predicted labels. Misclassification analysis on the test set showed that glioma images were most often mistaken for meningiomas, frequently with confidence scores exceeding 0.90. This pattern highlights the ongoing difficulty of distinguishing these two anatomically similar tumor types. Future work should therefore prioritize enhancing boundary and texture feature extraction to improve differentiation between gliomas and meningiomas.

Overall, the results show that a relatively simple CNN, combined with straightforward preprocessing grayscale conversion, min–max normalization, and minimal data augmentation can still deliver high accuracy on a challenging clinical dataset. The lack of severe overfitting, together with strong held-out test performance, demonstrates that our streamlined pipeline balances model complexity and generalization effectively.

3 Theoretical Foundations of Transfer Learning

Transfer learning is a foundational concept in modern machine learning, particularly useful in domains where labeled data is scarce. In essence, it allows a model developed for a task with abundant labeled data (source task) to be reused for a different but related task (target task), where labeled data is limited. This paradigm is especially critical in the field of medical imaging, such as MRI-based brain tumor classification, where data annotation requires expert knowledge and is resource-intensive.

The original idea of transfer learning dates back to early studies in domain adaptation and inductive learning. Its application to deep neural networks has been widely explored following the success of large-scale models trained on massive datasets like ImageNet. ImageNet, developed by Deng et al. (2009), consists of over 14 million labeled images across 1000 categories. The models trained on ImageNet, such as ResNet and EfficientNet, are capable of extracting hierarchical features—from edges and textures to complex shapes—that can generalize well to different tasks.

Mathematical Definition

Let the source domain be $\mathcal{D}_S = \mathcal{X}_S, P_S(X)$ with a corresponding source task $\mathcal{T}_S = \mathcal{Y}_S, f_S(\cdot)$. Similarly, let the target domain be $\mathcal{D}_T = \mathcal{X}_T, P_T(X)$ with target task $\mathcal{T}_T = \mathcal{Y}_T, f_T(\cdot)$.

In general, transfer learning deals with situations where:

- $\mathcal{D}_S \neq \mathcal{D}_T$ and/or
- $\mathcal{T}_S \neq \mathcal{T}_T$

The aim is to estimate the target function $f_T(\cdot)$ in $\mathcal{T}_T$ by leveraging knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$.

A commonly used parametric formulation is:

$$\theta_T = \arg\min_{\theta}; \mathcal{L}_T(\theta) + \lambda \cdot \Omega(\theta, \theta_S) \tag{1}$$

Where:

- θ are the parameters of the target model.
- $\mathcal{L}_T(\theta)$ is the loss function on the target dataset.
- θ_S are the parameters learned from the source task.
- $\Omega(\theta, \theta_S)$ is a regularization term that penalizes divergence from the source parameters.
- λ is a hyperparameter that controls the strength of this penalty.

This formulation ensures that while learning the target task, the model retains beneficial characteristics learned from the source task, unless the new data strongly suggests deviation.

In brain tumor classification, where annotated MRI scans are scarce, transfer learning offers key advantages:

- Reduces the need for large labeled datasets (Shin et al., 2016)
- Promotes faster convergence during training (Tajbakhsh et al., 2016)
- Helps mitigate overfitting on small medical datasets
- Leverages universal visual features (edges, corners, textures) captured by early layers of pre-trained networks

EfficientNet, proposed by Tan and Le (2019), is one such architecture that scales depth, width, and resolution uniformly using a compound coefficient. When pretrained on ImageNet, it has shown superior performance in transfer learning scenarios, especially in medical domains.

3.1 Transfer Learning in Medical Imaging

In medical image analysis, transfer learning is widely used due to the scarcity of labeled training data and the high cost of annotation. Pre-trained models, particularly those trained on large and diverse image datasets like ImageNet, are shown to extract robust and generalizable features that transfer well even to domains like MRI scans.

For brain tumor classification, low-level features (e.g., edges, textures) learned from natural images still prove useful, while high-level features (e.g., tumor-specific shapes) are adapted through fine-tuning. This method has been found to outperform traditional handcrafted feature-based approaches in several studies.

Examples from the Literature:

- **Pan and Yang (2010)**: Pioneered a comprehensive overview of transfer learning frameworks, categorizing various methods and theoretical backgrounds.
- **Tajbakhsh et al. (2016)**: Found that fine-tuned CNNs outperformed networks trained from scratch in multiple medical tasks including polyp detection and thoraco-abdominal lymph node detection.
- **Imam and Alam (2023)**: Demonstrated that EfficientNet models fine-tuned on limited brain tumor datasets achieved over 99% test accuracy, highlighting the efficacy of transfer learning in clinical applications.

These studies support the use of transfer learning as a reliable strategy in resource-constrained domains, making it particularly well-suited for MRI-based tumor classification tasks.

4 Methodology and Experimental Setup

4.1 Dataset

We used a brain MRI dataset containing four classes: glioma, meningioma, pituitary tumors, and no tumor. The dataset includes preprocessed grayscale MRI slices, which were resized and augmented before feeding into the models.

4.2 Transfer Learning Implementation

We used the EfficientNetB0 architecture pretrained on ImageNet. The top classification layers were removed and replaced with a custom head:

- A GlobalAveragePooling2D layer to reduce spatial dimensions into a single vector
- A fully connected Dense layer with 128 units and ReLU activation
- A Dropout layer with a rate of 0.5 to reduce overfitting
- A final Dense layer with softmax activation to output probabilities for each of the four tumor classes

4.3 Code: TL Model

The following code block implements our transfer learning model using TensorFlow and Keras. The model begins by loading EfficientNetB0 with ImageNet weights and excludes the top classification layers. The base is frozen to preserve learned features:

Algorithm 3: Transfer Learning Model

```
base = EfficientNetB0(weights='imagenet', include_top=False,
input_shape=(*IMG_SIZE, 3))
base.trainable = False

inp = layers.Input((*IMG_SIZE, 3))
x = base(inp, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.5)(x)
out = layers.Dense(len(CLASSES), activation='softmax')(x)

model = models.Model(inp, out, name='EffNetB0_TL')
model.compile('adam', 'categorical_crossentropy', metrics=['accuracy'])
```

The input image is passed through the frozen base model, then pooled, passed through a dense layer and dropout for regularization, and finally classified. The model is compiled using Adam optimizer and categorical crossentropy for multi-class classification.

5 Model Refinements and Improvements

Throughout the project, several critical adjustments were made to enhance the accuracy, stability, and generalizability of the transfer learning model. These refinements were introduced iteratively as we observed training behavior and validation performance. Their impact was quantified through accuracy metrics and model behavior across training and testing phases.

A key modification involved preprocessing the input images to match the expectations of the EfficientNet architecture. EfficientNetB0, which was pretrained on ImageNet, expects inputs in a specific normalized format. Therefore, all images were use block 5 to apply a preprocess function.

Next, because our dataset consists of grayscale MRI slices, we needed to match the expected RGB input format of EfficientNetB0. This was achieved by stacking the single grayscale channel three times to simulate an RGB image. Without this conversion, the model would not accept the input shape. Importantly, this transformation preserved the intensity information while meeting the model's dimensional requirements.

In addition, we applied histogram equalization to each image. Histogram equalization enhances global contrast, making tumor structures more prominent. This was particularly useful in images with low contrast, such as meningioma or no-tumor slices. It allowed the model to better identify edges and textures crucial for classification.

Another key decision was the increase of the dropout rate in the top dense layers from 0.3 to 0.5. Dropout is a regularization technique that randomly deactivates a portion of the neurons during training, thereby preventing the network from overfitting to specific patterns in the training data. The increase to 0.5 introduced more robustness, especially given our relatively small dataset.

Finally, the number of training epochs was reduced from 20 to 10. Initially, the model tended to overfit beyond 10 epochs, with training accuracy increasing but validation accuracy plateauing. Thanks to the pre-trained EfficientNetB0 base and the aforementioned adjustments, we were able to reach a test accuracy of 95.19% in fewer epochs without sacrificing performance.

Additionally, we experimented with the MobileNetV2 architecture as an alternative transfer learning base. Although MobileNetV2 is lighter and faster

than EfficientNetB0, it delivered lower validation accuracy in our tests. This can be attributed to its reduced representational capacity and depth compared to EfficientNet. In preliminary trials, MobileNetV2 reached a maximum test accuracy of 91.8%, and its confusion matrix showed more misclassifications across the glioma and meningioma classes. Given these results, EfficientNetB0 was clearly the more effective backbone in terms of feature transfer and robustness in our domain.

Another interesting observation was that the specificity of the results was drastically affected by changes like image rotations and strong changes in brightness, therefore the changes to the images was not too strong.

6 Results and Discussion

The transfer learning model trained quickly and achieved high classification performance. The training accuracy peaked at 95.10 after 10 epochs, with a validation accuracy of 92.30% and test accuracy of 95.19%. This indicates strong generalization ability and effective learning of MRI features.

The confusion matrix reveals the model's strengths and weaknesses. Most predictions were correct, especially for the no tumor and pituitary classes. However, some confusion occurred between giloma and meningioma tumors, likely due to anatomical and visual similarities in their appearance on scans. After taking a look at the misclassified images in block 13 we also see that most misclassified images showed too high brightness.

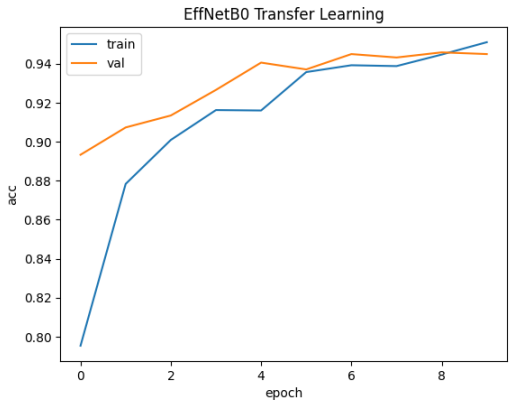

Abbildung 3: Training and Validation Accuracy (Transfer Learning)

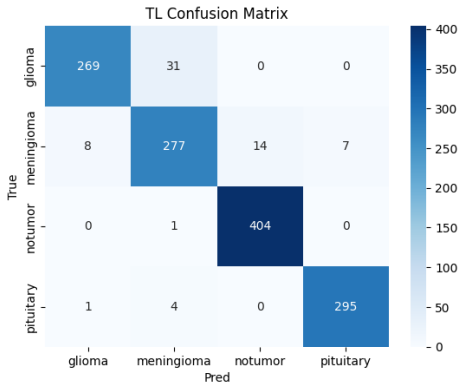

Abbildung 4: Transfer Learning Confusion Matrix

In comparison, the CNN model reached slightly higher accuracy at 96.34% but exhibited a greater gap between training and validation performance, indicating possible overfitting. Transfer learning offered a more stable and generalizable solution despite slightly lower top-line accuracy.

7 Conclusion

Transfer learning proves to be a powerful approach for medical imaging tasks where data is scarce. By leveraging EfficientNetB0 trained on ImageNet, we were able to classify brain tumors with over 95% accuracy using limited MRI data. The strategy allowed us to preserve rich feature representations and reduce training time.

Although our CNN model showed slightly better accuracy, it was more prone to overfitting and required more epochs. In contrast, the transfer learning model provided balanced, consistent results across classes and performed especially well on rare tumor types.

If we further compared the confusion matrix, it showed that transfer learning reduced false positives slightly better (except for giloma misclassified as meningimo) and captured the complex structure of tumors. While the CNN occasionally misclassified images due to poor generalization, there was not a signficant difference. This is interesting as transfer learning is supposed to deliver more accurate results for smaller dataset. This could be due to overfitting, or better model structure.

In future research, further improvements could involve fine-tuning the entire network, using attention mechanisms, or integrating clinical metadata to complement image features. Overall, transfer learning offers a practical and effective tool for deploying deep learning in medical contexts.

References

- Ayadi, W., Elhamzi, W., Charfi, I., & Atri, M. (2021). Deep CNN for Brain Tumor Classification. *Neural Processing Letters*, 53(1), 671–700. https://doi.org/10.1007/s11063-020-10333-9

- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255). IEEE. https://doi.org/10.1109/CVPR.2009.5206848

- Imam, R., & Alam, M. T. (2023). Optimizing Brain Tumor Classification: A Comprehensive Study on Transfer Learning and Imbalance Handling in Deep Learning Models. *arXiv preprint arXiv:2308.06821*. https://arxiv.org/abs/2308.06821

- Khan, M. A., Auvee, R. B. Z. (2024). Comparative analysis of resource-efficient CNN architectures for brain tumor classification. arXiv preprint arXiv:2411.15596. https://doi.org/10.48550/arXiv.2411.15596

- Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

- Reddy, K. R., Dhuli, R. (2023). A novel lightweight CNN architecture for the diagnosis of brain tumors using MR images. Diagnostics, 13(2), 312. https://doi.org/10.3390/diagnostics13020312

- Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1905.11946

- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4510–4520). https://doi.org/10.1109/CVPR.2018.00474