REPORT OF PRE-TRAINED MODEL

PyTorch

ÖZET

The report highlights the comparative analysis of machine learning models on image recognition tasks, emphasizing the critical balance between accuracy, computational efficiency, and model size necessary for optimal application performance.

Ali Huseynov

AI engineer...

https://pytorch.org/vision/stable/models.html

Weight	Acc@1	Acc@5	Params	GFLOPS	Recipe
AlexNet_Weights.IMAGENET1K_V1	56.522	79.066	61.1M	0.71	link
ConvNeXt_Base_Weights.IMAGENET1K_V1	84.062	96.87	88.6M	15.36	link
ConvNeXt_Large_Weights.IMAGENET1K_V1	84.414	96.976	197.8M	34.36	link
ConvNeXt_Small_Weights.IMAGENET1K_V1	83.616	96.65	50.2M	8.68	link
ConvNeXt_Tiny_Weights.IMAGENET1K_V1	82.52	96.146	28.6M	4.46	link
DenseNet121_Weights.IMAGENET1K_V1	74.434	91.972	8.0M	2.83	link
DenseNet161_Weights.IMAGENET1K_V1	77.138	93.56	28.7M	7.73	link
DenseNet169_Weights.IMAGENET1K_V1	75.6	92.806	14.1M	3.36	link
DenseNet201_Weights.IMAGENET1K_V1	76.896	93.37	20.0M	4.29	link
EfficientNet_B0_Weights.IMAGENET1K_V1	77.692	93.532	5.3M	0.39	link
EfficientNet_B1_Weights.IMAGENET1K_V1	78.642	94.186	7.8M	0.69	link
EfficientNet_B1_Weights.IMAGENET1K_V2	79.838	94.934	7.8M	0.69	link
EfficientNet_B2_Weights.IMAGENET1K_V1	80.608	95.31	9.1M	1.09	link
EfficientNet_B3_Weights.IMAGENET1K_V1	82.008	96.054	12.2M	1.83	link
EfficientNet_B4_Weights.IMAGENET1K_V1	83.384	96.594	19.3M	4.39	link
EfficientNet_B5_Weights.IMAGENET1K_V1	83.444	96.628	30.4M	10.27	link
EfficientNet_B6_Weights.IMAGENET1K_V1	84.008	96.916	43.0M	19.07	link
EfficientNet_B7_Weights.IMAGENET1K_V1	84.122	96.908	66.3M	37.75	link
EfficientNet_V2_L_Weights.IMAGENET1K_V1	85.808	97.788	118.5M	56.08	link
EfficientNet_V2_M_Weights.IMAGENET1K_V1	85.112	97.156	54.1M	24.58	link
EfficientNet_V2_S_Weights.IMAGENET1K_V1	84.228	96.878	21.5M	8.37	link
GoogLeNet_Weights.IMAGENET1K_V1	69.778	89.53	6.6M	1.5	link
Inception_V3_Weights.IMAGENET1K_V1	77.294	93.45	27.2M	5.71	link
MNASNet0_5_Weights.IMAGENET1K_V1	67.734	87.49	2.2M	0.1	link
MNASNet0_75_Weights.IMAGENET1K_V1	71.18	90.496	3.2M	0.21	link
MNASNet1_0_Weights.IMAGENET1K_V1	73.456	91.51	4.4M	0.31	link
MNASNet1_3_Weights.IMAGENET1K_V1	76.506	93.522	6.3M	0.53	link
MaxVit_T_Weights.IMAGENET1K_V1	83.7	96.722	30.9M	5.56	link
MobileNet_V2_Weights.IMAGENET1K_V1	71.878	90.286	3.5M	0.3	link
MobileNet_V2_Weights.IMAGENET1K_V2	72.154	90.822	3.5M	0.3	link
MobileNet_V3_Large_Weights.IMAGENET1K_V1	74.042	91.34	5.5M	0.22	link
MobileNet_V3_Large_Weights.IMAGENET1K_V2	75.274	92.566	5.5M	0.22	link
MobileNet_V3_Small_Weights.IMAGENET1K_V1	67.668	87.402	2.5M	0.06	link
RegNet_X_16GF_Weights.IMAGENET1K_V1	80.058	94.944	54.3M	15.94	link
RegNet_X_16GF_Weights.IMAGENET1K_V2	82.716	96.196	54.3M	15.94	link
RegNet_X_1_6GF_Weights.IMAGENET1K_V1	77.04	93.44	9.2M	1.6	link
RegNet_X_1_6GF_Weights.IMAGENET1K_V2	79.668	94.922	9.2M	1.6	link
RegNet_X_32GF_Weights.IMAGENET1K_V1	80.622	95.248	107.8M	31.74	link
RegNet_X_32GF_Weights.IMAGENET1K_V2	83.014	96.288	107.8M	31.74	link
RegNet_X_3_2GF_Weights.IMAGENET1K_V1	78.364	93.992	15.3M	3.18	link
RegNet_X_3_2GF_Weights.IMAGENET1K_V2	81.196	95.43	15.3M	3.18	link
RegNet_X_400MF_Weights.IMAGENET1K_V1	72.834	90.95	5.5M	0.41	link
RegNet_X_400MF_Weights.IMAGENET1K_V2	74.864	92.322	5.5M	0.41	link
RegNet_X_800MF_Weights.IMAGENET1K_V1	75.212	92.348	7.3M	0.8	link
RegNet_X_800MF_Weights.IMAGENET1K_V2	77.522	93.826	7.3M	0.8	link
RegNet_X_8GF_Weights.IMAGENET1K_V1	79.344	94.686	39.6M	8	link
RegNet_X_8GF_Weights.IMAGENET1K_V2	81.682	95.678	39.6M	8	link
RegNet_Y_128GF_Weights.IMAGENET1K_SWAG_E2E_V1	88.228	98.682	644.8M	374.57	link
RegNet_Y_128GF_Weights.IMAGENET1K_SWAG_LINEAR_V1	86.068	97.844	644.8M	127.52	link
RegNet_Y_16GF_Weights.IMAGENET1K_V1	80.424	95.24	83.6M	15.91	link
RegNet_Y_16GF_Weights.IMAGENET1K_V2	82.886	96.328	83.6M	15.91	link
RegNet_Y_16GF_Weights.IMAGENET1K_SWAG_E2E_V1	86.012	98.054	83.6M	46.73	link

Weight	Acc@1	Acc@5	Params	GFLOPS	Recipe
RegNet_Y_16GF_Weights.IMAGENET1K_SWAG_LINEAR_V1	83.976	97.244	83.6M	15.91	link
RegNet_Y_1_6GF_Weights.IMAGENET1K_V1	77.95	93.966	11.2M	1.61	link
RegNet_Y_1_6GF_Weights.IMAGENET1K_V2	80.876	95.444	11.2M	1.61	link
RegNet_Y_32GF_Weights.IMAGENET1K_V1	80.878	95.34	145.0M	32.28	link
RegNet_Y_32GF_Weights.IMAGENET1K_V2	83.368	96.498	145.0M	32.28	link
RegNet_Y_32GF_Weights.IMAGENET1K_SWAG_E2E_V1	86.838	98.362	145.0M	94.83	link
RegNet_Y_32GF_Weights.IMAGENET1K_SWAG_LINEAR_V1	84.622	97.48	145.0M	32.28	link
RegNet_Y_3_2GF_Weights.IMAGENET1K_V1	78.948	94.576	19.4M	3.18	link
RegNet_Y_3_2GF_Weights.IMAGENET1K_V2	81.982	95.972	19.4M	3.18	link
RegNet_Y_400MF_Weights.IMAGENET1K_V1	74.046	91.716	4.3M	0.4	link
RegNet_Y_400MF_Weights.IMAGENET1K_V2	75.804	92.742	4.3M	0.4	link
RegNet_Y_800MF_Weights.IMAGENET1K_V1	76.42	93.136	6.4M	0.83	link
RegNet_Y_800MF_Weights.IMAGENET1K_V2	78.828	94.502	6.4M	0.83	link
RegNet_Y_8GF_Weights.IMAGENET1K_V1	80.032	95.048	39.4M	8.47	link
RegNet_Y_8GF_Weights.IMAGENET1K_V2	82.828	96.33	39.4M	8.47	link
ResNeXt101_32X8D_Weights.IMAGENET1K_V1	79.312	94.526	88.8M	16.41	link
ResNeXt101_32X8D_Weights.IMAGENET1K_V2	82.834	96.228	88.8M	16.41	link
ResNeXt101_64X4D_Weights.IMAGENET1K_V1	83.246	96.454	83.5M	15.46	link
ResNeXt50_32X4D_Weights.IMAGENET1K_V1	77.618	93.698	25.0M	4.23	link
ResNeXt50_32X4D_Weights.IMAGENET1K_V2	81.198	95.34	25.0M	4.23	link
ResNet101_Weights.IMAGENET1K_V1	77.374	93.546	44.5M	7.8	link
ResNet101_Weights.IMAGENET1K_V2	81.886	95.78	44.5M	7.8	link
ResNet152_Weights.IMAGENET1K_V1	78.312	94.046	60.2M	11.51	link
ResNet152_Weights.IMAGENET1K_V2	82.284	96.002	60.2M	11.51	link
ResNet18_Weights.IMAGENET1K_V1	69.758	89.078	11.7M	1.81	link
ResNet34_Weights.IMAGENET1K_V1	73.314	91.42	21.8M	3.66	link
ResNet50_Weights.IMAGENET1K_V1	76.13	92.862	25.6M	4.09	link
ResNet50_Weights.IMAGENET1K_V2	80.858	95.434	25.6M	4.09	link
ShuffleNet_V2_X0_5_Weights.IMAGENET1K_V1	60.552	81.746	1.4M	0.04	link
ShuffleNet_V2_X1_0_Weights.IMAGENET1K_V1	69.362	88.316	2.3M	0.14	link
ShuffleNet_V2_X1_5_Weights.IMAGENET1K_V1	72.996	91.086	3.5M	0.3	link
ShuffleNet_V2_X2_0_Weights.IMAGENET1K_V1	76.23	93.006	7.4M	0.58	link
SqueezeNet1_0_Weights.IMAGENET1K_V1	58.092	80.42	1.2M	0.82	link
SqueezeNet1_1_Weights.IMAGENET1K_V1	58.178	80.624	1.2M	0.35	link
Swin_B_Weights.IMAGENET1K_V1	83.582	96.64	87.8M	15.43	link
Swin_S_Weights.IMAGENET1K_V1	83.196	96.36	49.6M	8.74	link
Swin_T_Weights.IMAGENET1K_V1	81.474	95.776	28.3M	4.49	link
Swin_V2_B_Weights.IMAGENET1K_V1	84.112	96.864	87.9M	20.32	link
Swin_V2_S_Weights.IMAGENET1K_V1	83.712	96.816	49.7M	11.55	link
Swin_V2_T_Weights.IMAGENET1K_V1	82.072	96.132	28.4M	5.94	link
VGG11_BN_Weights.IMAGENET1K_V1	70.37	89.81	132.9M	7.61	link
VGG11_Weights.IMAGENET1K_V1	69.02	88.628	132.9M	7.61	link
VGG13_BN_Weights.IMAGENET1K_V1	71.586	90.374	133.1M	11.31	link
VGG13_Weights.IMAGENET1K_V1	69.928	89.246	133.0M	11.31	link
VGG16_BN_Weights.IMAGENET1K_V1	73.36	91.516	138.4M	15.47	link
VGG16_Weights.IMAGENET1K_V1	71.592	90.382	138.4M	15.47	link
VGG16_Weights.IMAGENET1K_FEATURES	nan	nan	138.4M	15.47	link
VGG19_BN_Weights.IMAGENET1K_V1	74.218	91.842	143.7M	19.63	link
VGG19_Weights.IMAGENET1K_V1	72.376	90.876	143.7M	19.63	link
ViT_B_16_Weights.IMAGENET1K_V1	81.072	95.318	86.6M	17.56	link
ViT_B_16_Weights.IMAGENET1K_SWAG_E2E_V1	85.304	97.65	86.9M	55.48	link
ViT_B_16_Weights.IMAGENET1K_SWAG_LINEAR_V1	81.886	96.18	86.6M	17.56	link
ViT_B_32_Weights.IMAGENET1K_V1	75.912	92.466	88.2M	4.41	link
ViT_H_14_Weights.IMAGENET1K_SWAG_E2E_V1	88.552	98.694	633.5M	1016.72	link
ViT_H_14_Weights.IMAGENET1K_SWAG_LINEAR_V1	85.708	97.73	632.0M	167.29	link
ViT_L_16_Weights.IMAGENET1K_V1	79.662	94.638	304.3M	61.55	link

Weight	Acc@1	Acc@5	Params	GFLOPS	Recipe
ViT_L_16_Weights.IMAGENET1K_SWAG_E2E_V1	88.064	98.512	305.2M	361.99	link
ViT_L_16_Weights.IMAGENET1K_SWAG_LINEAR_V1	85.146	97.422	304.3M	61.55	link
ViT_L_32_Weights.IMAGENET1K_V1	76.972	93.07	306.5M	15.38	link
Wide_ResNet101_2_Weights.IMAGENET1K_V1	78.848	94.284	126.9M	22.75	link
Wide_ResNet101_2_Weights.IMAGENET1K_V2	82.51	96.02	126.9M	22.75	link
Wide_ResNet50_2_Weights.IMAGENET1K_V1	78.468	94.086	68.9M	11.4	link
Wide_ResNet50_2_Weights.IMAGENET1K_V2	81.602	95.758	68.9M	11.4	link

The examination of six cutting-edge machine learning models——reveals a nuanced landscape of options available for image classification tasks, each balancing computational efficiency, model size, and accuracy in unique ways.

Three Python files were uploaded to GitHub, each designed to fulfill distinct tasks within a machine learning or data analysis workflow:

1. **5Model.py**: This script is tailored for comparing five different models, showcasing their capabilities by providing the top five predictions from each. It's a comparative tool that allows for an in-depth evaluation of how each model performs on the same dataset or input, highlighting the prediction diversity among different models.
2. **Combined.py**: Engineered to operate with the same five models, this script not only compares the results but also delves into the accuracy percentages and detailed predictions of each model. It offers a nuanced view by revealing which models perform better in terms of prediction accuracy and provides full insights into the predictions made by each model.
3. **ForAPI.py**: Shares a core codebase with Combined.py, but is streamlined for a different output format. While Combined.py presents model names alongside their predictions, ForAPI.py is designed to output only the predictions and their confidence percentages, omitting the model names. This simplification is particularly suited for API responses where the emphasis is on the prediction details rather than the specifics of the model making those predictions.

You can find classes in this link:

https://github.com/raghakot/keras-vis/blob/master/resources/imagenet_class_index.json

These are 10 links of pictures that I used my codes, and these are results:

https://m.media-amazon.com/images/I/71KwPy8BPiL._AC_UF1000,1000_QL80_.jpg https://media.istockphoto.com/id/155439315/photo/passenger-airplane-flying-above-clouds-during-sunset.jpg?s=612x612&w=0&k=20&c=LJWadbs3B-jSGJBVy9s0f8gZMHi2NvWFXa3VJ2lFcL0=

https://res.cloudinary.com/padi/image/upload/v1701290031/CDN/commerce/wreck-diver.jpg

https://c1.wallpaperflare.com/preview/826/375/678/race-car-sport-car-sport-speed.jpg

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQEysfzzeDbVyV9s9Atx16tFld4snlaCbvGFg&usqp=CAU

https://avatars.mds.yandex.net/i?id=7be615771fa05f41231c53d56cefa446eda3f067-10178110-images-thumbs&n=13

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQEysfzzeDbVyV9s9Atx16tFld4snlaCbvGFg&usqp=CAU

https://idfg.idaho.gov/sites/default/files/styles/article_header/public/field/image/34a99d1c-75c1-4035-8dbe-76546f83b279.jpg?itok=OCpTFYy6

https://www.freecodecamp.org/news/content/images/2021/11/niclas-illg-wzVQp_NRIHg-unsplash.jpg

https://images2.minutemediacdn.com/image/upload/c_fill,w_1440,ar_16:9,f_auto,q_auto,g_auto/shape/cover/sport/mirror-gettyimages-537282880-0-c1e4c2d1b9d7ff400d4cc8fce2301f43.jpg

https://smarthistory.org/wp-content/uploads/2019/07/cordoba.jpg

These are 10 links of pictures that I used my codes, and these are results:

Model Overview:

EfficientNetV2 is an evolution of the EfficientNet architecture that prioritizes faster training speed and greater model efficiency. The "V2" versions introduce improvements such as progressive learning techniques and enhanced scaling methods. The specific variant **EfficientNet_V2_S** represents a small-sized model within the V2 family. Despite its size, it aims to deliver a balance between speed and accuracy, optimizing it for performance on a wide range of computing platforms.

ConvNeXt_Tiny

ConvNeXt represents a new family of Convolutional Neural Networks (CNNs) that incorporates design principles from Transformers. The ConvNeXt models have been redesigned architecture-wise, with simplified designs but more effective scaling strategies compared to previous convolutional models. The "Tiny" variant is designed to be smaller in parameter count and computational complexity, offering an efficient option while still maintaining high accuracy.

Swin_Transformer_Tiny

Swin Transformer is a type of Transformer model tailored specifically for vision tasks. It introduces a hierarchical Transformer architecture with shift windows, enabling efficient modeling of image data. The unique aspect of Swin Transformers is their ability to model at various scales and their computational efficiency, which scales linearly with image size. The "Tiny" variant is the smallest amongst its family, designed to deliver solid performance with minimal computational overhead.

MobileNetV2

MobileNetV2 is part of the MobileNets family, which are lightweight deep neural networks designed for mobile and resource-constrained environments. It introduces the concept of inverted residuals and linear bottlenecks to encapsulate the model's layers, focusing on optimizing the speed and efficiency of mobile vision applications. MobileNetV2 strikes a balance between performance and size, making it suitable for a variety of real-time applications on mobile devices.

ResNet50

ResNet50 is a variant of the Residual Networks (ResNets), a highly popular architecture due to its deep structure that can go up to hundreds of layers. It utilizes skip connections, or shortcuts to jump over some layers. The "50" refers to the fact that it contains 50 layers deep. ResNets, and particularly ResNet50, are widely used for a variety of vision tasks because they allow for training very deep neural networks without a significant increase in the vanishing gradient problem.

Addendum to Image Classification Model Analysis Report: Ali's Recommendation:

Based on the provided analysis of images using five different model architectures (EfficientNet_V2_S, ConvNeXt_Tiny, Swin_Transformer_Tiny, MobileNetV2, and ResNet50), we can glean significant insights into each model's performance, utility, and storage considerations. Below is a summary report that outlines the findings and concludes with recommendations based on your priorities: accuracy versus storage efficiency.

Summary of Model Performances

- **EfficientNet_V2_S** generally showed good performance across various types of images, making relatively accurate classifications. It was particularly strong in scenarios where fine-grained detail was essential, like in distinguishing between different types of balls or identifying specific computer parts.
- **ConvNeXt_Tiny** performed commendably, often close to or slightly better than EfficientNet_V2_S in certain tasks. Its strengths were evident in clearly identifiable objects but sometimes made less precise predictions on more complex scenes.
- **Swin_Transformer_Tiny** demonstrated excellent capability, particularly in correctly identifying diverse classes with high confidence. It excelled in scenarios requiring an understanding of complex structures and details, reflecting its design's emphasis on capturing hierarchical features.
- **MobileNetV2** stands out for its storage and computational efficiency, making it an excellent choice for applications with strict resource constraints. While its accuracy was generally lower compared to the more sophisticated models, MobileNetV2 still provided very reliable classifications, especially considering its relatively small size and lower computational demand.
- **ResNet50** showed high accuracy across a broad range of images, making it the top performer in many cases. Its ability to discern fine details and understand complex scenes validates its widespread adoption in the deep learning community. ResNet50 combines depth and efficiency, balancing computational and storage demands with high-performance metrics.

Recommendations

- **For Maximum Accuracy**: If the primary goal is achieving the highest accuracy without significant constraints on computational resources and storage, **ResNet50** is the best choice. Its consistent performance across various image types and complexities makes it a robust option for tasks where precision is crucial.

- **For Balanced Performance and Efficiency**: **EfficientNet_V2_S** and **Swin_Transformer_Tiny** offer a good balance between model size and accuracy. These models are suitable for scenarios where a slight trade-off in performance for efficiency is acceptable. Between the two, **Swin_Transformer_Tiny** may have an edge in handling complex images, while **EfficientNet_V2_S** could be slightly more efficient storage-wise.
- **For Limited Storage and Computational Resources**: **MobileNetV2** is the go-to choice for environments where storage and computational efficiency are paramount. Its performance, while not at the pinnacle, is surprisingly good for its size, making it an excellent choice for mobile applications, embedded systems, or any application where resources are a constraint.

Conclusion

The decision on which model to adopt depends heavily on the specific needs and constraints of your application. If accuracy is non-negotiable, ResNet50 provides the best overall performance. For applications where every kilobyte matters or where computational power is limited, MobileNetV2 offers an admirable balance between efficiency and effectiveness. For use cases that lie in the middle, EfficientNet_V2_S and Swin_Transformer_Tiny stand as viable options that balance the trade-offs between size and accuracy.

In applications where changing requirements or scalability is a concern, employing a more adaptive approach—such as starting with MobileNetV2 for rapid prototyping or applications with strict resource limitations and then transitioning to ResNet50 as conditions allow—could provide a strategic advantage.