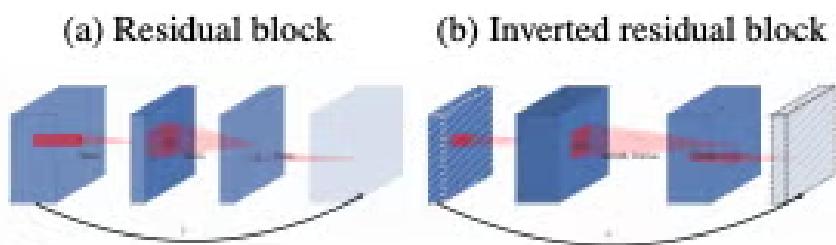


[Learn](#)[Community](#)[Projects](#)[Docs](#)[Blog & News](#)[About](#)[JOIN](#)

## Model Hub

# MobileNet v2



Input	Operator	<i>t</i>	<i>c</i>	<i>n</i>
$224^2 \times 3$	conv2d	-	32	1
$112^2 \times 32$	bottleneck	1	16	1
$112^2 \times 16$	bottleneck	6	24	2
$56^2 \times 24$	bottleneck	6	32	3
$28^2 \times 32$	bottleneck	6	64	4
$14^2 \times 64$	bottleneck	6	96	3
$14^2 \times 96$	bottleneck	6	160	3
$7^2 \times 160$	bottleneck	6	320	1
$7^2 \times 320$	conv2d 1x1	-	1280	1
$7^2 \times 1280$	avgpool 7x7	-	-	1
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-

```
import torch
model = torch.hub.load('pytorch/vision:v0.10.0', 'mobilenet_v2', pretrain=True)
model.eval()
```

This website utilizes technologies such as cookies to enable essential site functionality, as well as for analytics, personalization, and targeted advertising. [Privacy Policy](#)

[Accept](#)[Deny Non-Essential](#)[Manage Preferences](#)

[Learn](#)[Community](#)[Projects](#)[Docs](#)[Blog & News](#)[About](#)[JOIN](#)

```
# sample execution (requires torchvision)
from PIL import Image
from torchvision import transforms
input_image = Image.open(filename)
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.
])
input_tensor = preprocess(input_image)
input_batch = input_tensor.unsqueeze(0) # create a mini-batch as expected by the model

# move the input and model to GPU for speed if available
if torch.cuda.is_available():
    input_batch = input_batch.to('cuda')
    model.to('cuda')

with torch.no_grad():
    output = model(input_batch)
```

This website utilizes technologies such as cookies to enable essential site functionality, as well as for analytics, personalization, and targeted advertising. [Privacy Policy](#)

[Learn](#) [Community](#) [Projects](#) [Docs](#) [Blog & News](#) [About](#) [JOIN](#)

```
# Show top categories per image
top5_prob, top5_catid = torch.topk(probabilities, 5)
for i in range(top5_prob.size(0)):
    print(categories[top5_catid[i]], top5_prob[i].item())
```

## Model Description

The MobileNet v2 architecture is based on an inverted residual structure where the input and residual block are thin bottleneck layers opposite to traditional residual models which use exp representations in the input. MobileNet v2 uses lightweight depthwise convolutions to filter fea intermediate expansion layer. Additionally, non-linearities in the narrow layers were removed i representational power.

Model structure	Top-1 error	Top-5 error
mobilenet_v2	28.12	9.71

This website utilizes technologies such as cookies to enable essential site functionality, as well as for analytics, personalization, and targeted advertising. [Privacy Policy](#)

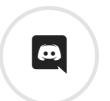
[Learn](#)[Community](#)[Projects](#)[Docs](#)[Blog & News](#)[About](#)[JOIN](#)

Submitted by the PyTorch team

[View on GitHub 17.4k](#)

[Open on Google Collab](#)

[Open Model Demo](#)



© 2025 PyTorch. Copyright © The Linux Foundation®. All rights reserved. The Linux Foundation has registered trademarks and uses trademarks. For more information, including terms of use, privacy policy, and trademark usage, please see our [Policies page](#). [Trademark Usage](#). [Privacy Policy](#).

This website utilizes technologies such as cookies to enable essential site functionality, as well as for analytics, personalization, and targeted advertising. [Privacy Policy](#)