Table 3. Human and model multi-label accuracy on three subsets of the ImageNet and ImageNetV2 test sets. These results suggest that human labelers have an easier time identifying objects than dogs and organisms.

ImageNet multi-label accuracy (%)								
	All Images		**Dogs**		**Animals without dogs**		**Objects Only**	
Participant	Original	V2	Original	V2	Original	V2	Original	V2
resnet50	84.2	75.7	78.8	67.8	90.4	84.0	82.5	72.8
AdvProp	93.6	88.3	89.8	80.0	97.4	93.6	92.3	86.7
FixResNeXt	95.5	89.6	92.4	79.1	97.4	93.6	95.0	89.1
Human A	91.9	91.1	74.5	73.9	89.4	86.9	97.0	96.7
Human B	94.7	93.9	78.8	78.2	94.2	92.6	98.3	97.8
Human C	96.2	96.7	80.5	82.6	97.1	96.4	99.1	99.8
Human D	95.7	94.8	83.9	80.8	94.5	93.0	98.8	98.4
Human E	97.2	96.5	86.4	90.4	98.7	97.7	98.8	97.0

Table 4. Human and model `top-1` accuracy on three subsets of the ImageNet and ImageNetV2 test sets. The values shown in this table suggest that human labelers have an easier time identifying objects than dogs and organisms. Moreover, human labelers are highly accurate on images on which they spent little time to assign a label.

ImageNet Top-1 accuracy (%)								
	All Images		**Without Dogs**		**Objects Only**		**Fast Images**	
Participant	Original	V2	Original	V2	Original	V2	Original	V2
resnet50	78.1	68.8	78.4	69.0	74.6	62.4	80.8	72.0
AdvProp	88.2	81.5	88.2	81.8	84.9	77.1	90.1	83.3
FixResNeXt	89.4	82.7	89.5	83.4	87.5	79.1	90.5	84.2
Human A	76.9	71.8	79.8	74.7	79.4	72.6	83.5	78.4
Human B	79.1	76.2	82.7	80.0	80.6	77.2	84.4	81.5
Human C	80.7	78.0	84.5	82.1	82.8	79.5	85.7	83.1
Human D	83.5	79.2	85.7	81.7	84.0	79.1	88.8	83.2
Human E	90.3	85.7	91.6	85.9	89.9	81.9	93.0	87.8

A. Accuracies on three disjoint subsets

To gain further insights in the capabilities of both machine and human labelers we compute their accuracies on three disjoint sets of classes: dogs, animals without dogs, and inanimate objects. The results can be found in Table 3

B. Top-1 Accuracies

In the main text, we measured the *multi-label* accuracy of both models and humans on both the ImageNet and ImageNetV2 test sets. For completeness, in Table 4 we now provide both the *top-1* and *multi-label* accuracy of models and humans on the same test sets. Figure 6 shows the scatters plot of `top-1`accuracies on ImageNet and ImageNetV2 test sets.

We note that under the `top-1`metric, we observe a substantially larger median accuracy drop of 4.3% between ImageNet and ImageNetV2 when compared to a median accuracy drop of 0.8% between the two datasets under the multi-label metric. As shown in Table 4, a similar accuracy drop accuracy drop exists on all three subsets studied in the main text.

While in Section 3 we address major issues with `top-1`accuracy for human evaluation, it is nonetheless interesting that such humans exhibit such a large performance drop in the `top-1`metric. In addition to the reasons mentioned in Section 3, we investigate two additional conjectures for the difference between `top-1`and multi-label result:

1. **Escape Hatches.** One potential failure mode of the multi-label metric would be an excess of images where the correct human prediction is a small or common object that is present in the scene but is presumably not the intended subject of the image. We denote these class labels "escape-hatch" labels as they allow the human to punt on a difficult

Table 5. Human and model multi-label accuracy on ImageNet and ImageNetV2 test sets after escape hatch imputation. We also provide the original accuracies for reference.

ImageNet multi-label accuracy (%)				
	With Escape Hatch Imputation		**Without Escape Hatch Imputation**	
Participant	Original	V2	Original	V2
Human A	82.7	79.8	91.9	91.1
Human B	87.1	84.1	94.7	93.9
Human C	91.1	91.6	96.2	96.7
Human D	87.5	84.0	95.7	94.8
Human E	88.0	86.6	97.2	96.5

classification task such as the difference between a `French bull dog` vs a `Boston terrier` to classify an easier background object such as `pole`.

Since the notion of escape-hatches could substantially inflate accuracies on the classification task, all human subjects provided alternative labels for any image for which they believed they used an escape hatch. To preserve our blind analysis this was done *before* the subjects viewed the true labels for the images. While this process isn't perfect as it relied on each subjects own judgement on what constitutes an escape-hatch, we believe this is unavoidable since the notion of an escape hatch label is highly subjective in nature.

In table 5 we present accuracies on the predictions with the escape hatch alternatives imputed, that is for each image the subject marked as an escape hatch we replaced his or her prediction with the secondary prediction provided. We note these numbers vary significantly across each of the participants as the set of escape hatch images for each participants varies. We see that this induces an approximately 3% accuracy drop in 4/5 participants.

2. **Multi-label proportion in ImageNet vs ImageNet V2.** Another possible explanation for the `top-1`accuracy discrepancy between ImageNet and ImageNetV2 could be a higher proportion of images with multiple labels in ImageNetV2. Among the 1000 images labeled, 30.0% (292/984) images in ImageNet contained multiple labels compared to 34.4% (337/980) images in ImageNetV2.

   If humans have a significantly lower `top-1` accuracy on images with multiple semantically correct labels, the higher proportion of multi-label images in ImageNetV2 could partially explain the accuracy drop between the two datasets. Figure 7 illustrates precisely this notion, we measure the `top-1` accuracy on two mutually exclusive subsets of ImageNet and ImageNetV2, those with exactly one correct label, and those with multiple correct labels. We find that human accuracy can degrade over 40% between images with a single correct label and those with multiple correct labels.

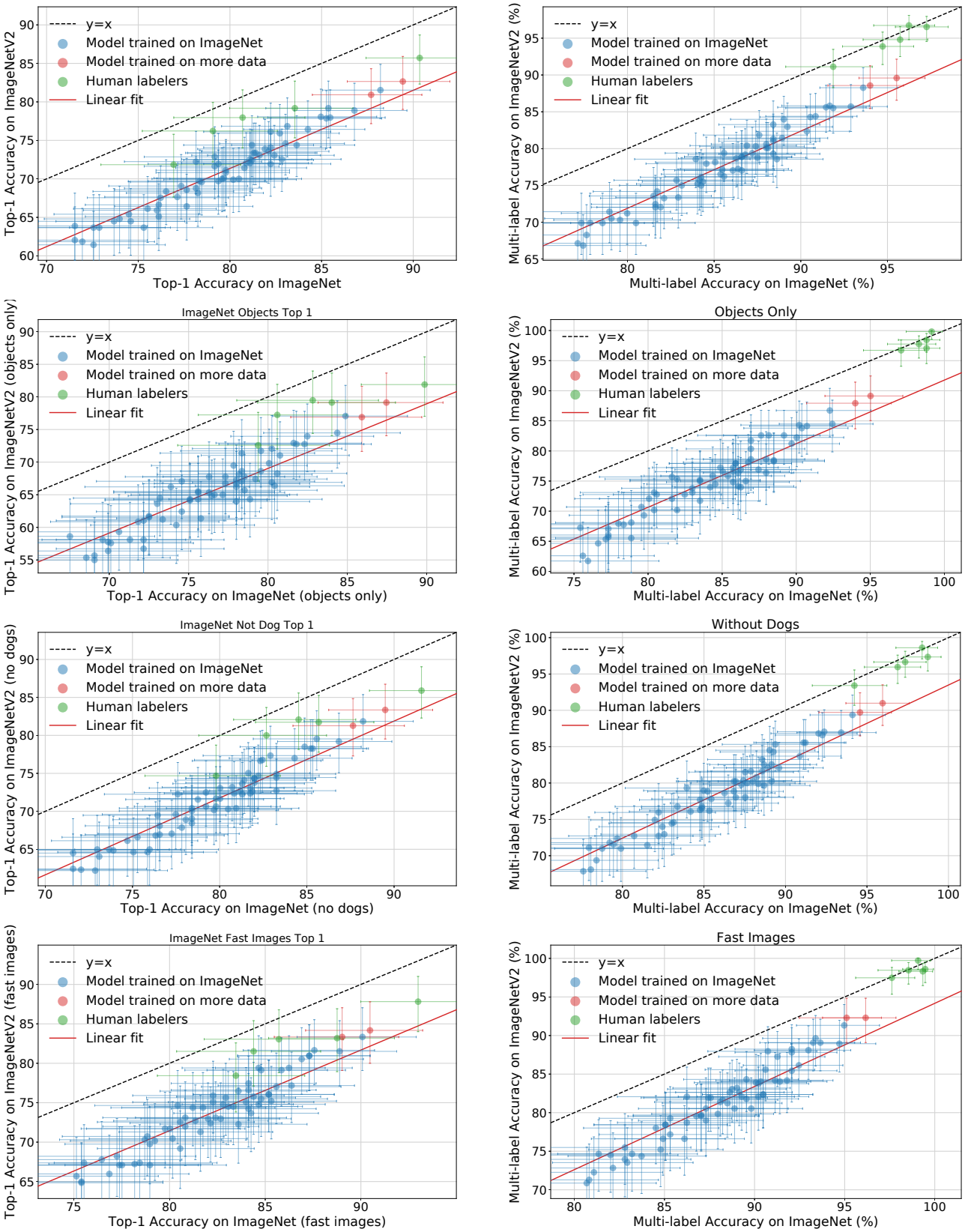

Figure 6. (left)`top-1`accuracies and (b) multi-label accuracies for both convolutional neural networks and five human labelers on the ImageNet validation set versus their accuracies on the ImageNetV2 test set. The confidence intervals are 95% Clopper-Pearson confidence intervals.

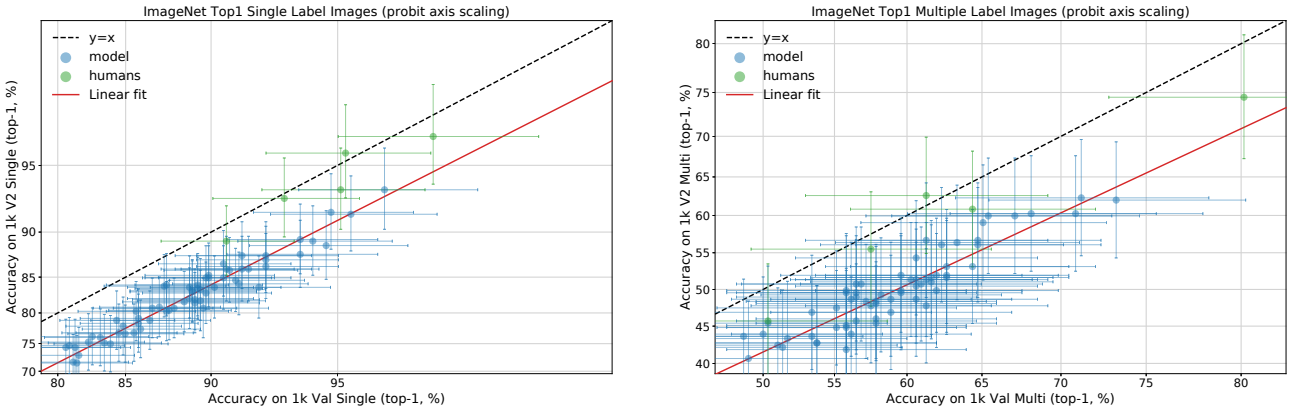

Figure 7. Scatter plot of `top-1`accuracies on subsets of ImageNet and ImageNetV2 with (left) A single "correct label (right) multiple "correct" labels. Note humans do *significantly* worse on images with multiple labels. The confidence intervals are 95% Clopper-Pearson confidence intervals.

C. Examples of training effective Images

(a) (b) (c)

Figure 8. Examples of ImageNet classes that are difficult to distinguish or identify for untrained human labelers. a) A `French bull dog` (top) versus a `Boston bull terrier` (bottom). Though the dogs appear similar, there are key differences between the two breeds. `French bulldogs` have a more muscular build and weigh more than `Boston terriers`. b) A `dragonfly` (top) versus a `damselfly` (bottom). Again, the classes may initially look similar, but dragonflies have wings perpendicular to their body at rest while damselflies have wings parallel to their body at rest. c) A `fire screen` (top) and an industrial `can opener` (bottom). Both images are not prototypical examples for their class. Hence knowing the class hierarchy in detail to recognize the most likely classes is helpful.

D. Time Spent Per Image

Figure 9 details the median time spent by a human on each image. We note all the time measured is *active* time the kparticipants spent searching the UI for a potential label.

E. Problematic Image removal

One key step of the initial annotation procedure was removing *problematic images*. An image was problematic if any of the below criteria held:

- The original ImageNet label (`top-1`label) was incorrect
- Image was a cartoon or drawing
- Image was excessively edited
- Image had inappropriate content

Out of the $40{,}683$ reviewed there were a total of 1206 images from ImageNet that were marked problematic and 686 images from ImageNetV2 marked problematic. In Table 6 we compute multi-label accuracies on both the problematic and non problematic subsets. Note accuracies are *substantially* lower on the problematic subset. Curiously accuracies are slightly higher on the problematic images from ImageNetV2 compared to ImageNet, we believe this is due to subjective decisions during the problematic image removal process and not some fundamental phenomena.

F. References for models in our testbed.

The following list contains all models we included in out testbed with references and links to the corresponding source code.

1. `alexnet` (Krizhevsky et al., 2012) `https://github.com/Cadene/pretrained-models.pytorch`

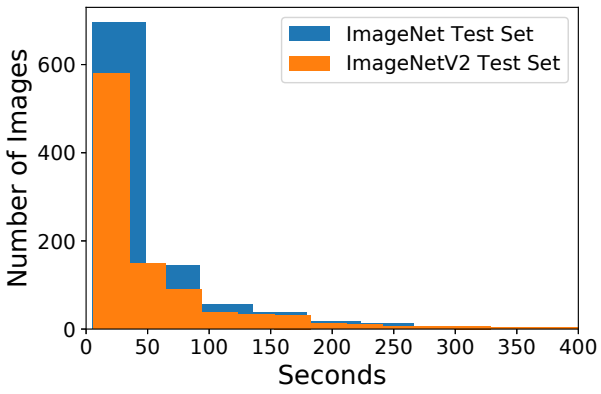

Figure 9. A histogram of the median time, measured in seconds, spent by the human labelers on images. We omitted are 24 outliers for which the median time spent by the human labelers was longer than 400 seconds: 13 outliers from the ImageNet test set and 11 outliers from the ImageNetV2 test set.

Table 6. Model multi-label accuracies on problematic and non-problematic subsets

ImageNet multi-label accuracy (%)				
	Non Problematic Images		**Problematic Images**	
Model	Original	V2	Original	V2
`resnet50`	84.2	74.0	66.6	68.4
`AdvProp`	93.7	87.7	72.1	73.1
`FixRes`	94.7	90.2	74.0	75.7

2. `advprop` (Xie et al., 2019a) `https://github.com/rwightman/pytorch-image-models`
3. `bninception` (Ioffe & Szegedy, 2015) `https://github.com/Cadene/pretrained-models.pytorch`
4. `cafferesnet101` (He et al., 2016a) `https://github.com/Cadene/pretrained-models.pytorch`
5. `densenet121` (Huang et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
6. `densenet161` (Huang et al., 2017)`https://github.com/Cadene/pretrained-models.pytorch`
7. `densenet169` (Huang et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
8. `densenet201` (Huang et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
9. `dpn107` (Chen et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
10. `dpn131` (Chen et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
11. `dpn68b` (Chen et al., 2017)`https://github.com/Cadene/pretrained-models.pytorch`
12. `dpn68` (Chen et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
13. `dpn92` (Chen et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
14. `dpn98` (Chen et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
15. `efficientnet_b7` (Tan & Le, 2019) `https://github.com/rwightman/pytorch-image-models`
16. `fbresnet152` (He et al., 2016a) `https://github.com/tensorflow/models/tree/master/research/slim/`
17. `fixresnext` (Touvron et al., 2019) `https://github.com/facebookresearch/FixRes`

18. `fv_4k` (Clinchant et al., 2007; Perronnin et al., 2010) `https://github.com/modestyachts/nondeep` FisherVector model using SIFT, local color statistic features, and 16 GMM centers.
19. `fv_16k` (Clinchant et al., 2007; Perronnin et al., 2010) `https://github.com/modestyachts/nondeep` FisherVector model using SIFT, local color statistic features, and 64 GMM centers.
20. `fv_64k` (Clinchant et al., 2007; Perronnin et al., 2010) `https://github.com/modestyachts/nondeep` FisherVector model using SIFT, local color statistic features, and 256 GMM centers.
21. `inception_resnet_v2_tf` (Szegedy et al., 2017) `https://github.com/tensorflow/models/tree/master/research/slim/`
22. `inception_v1_tf` (Szegedy et al., 2015) `https://github.com/tensorflow/models/tree/master/research/slim/`
23. `inception_v2_tf` (Ioffe & Szegedy, 2015) `https://github.com/tensorflow/models/tree/master/research/slim/`
24. `inception_v3_tf` (Szegedy et al., 2016) `https://github.com/tensorflow/models/tree/master/research/slim/`
25. `inception_v3` (Szegedy et al., 2016) `https://github.com/Cadene/pretrained-models.pytorch`
26. `inception_v4_tf` (Szegedy et al., 2017) `https://github.com/tensorflow/models/tree/master/research/slim/`
27. `inceptionresnetv2` (Ioffe & Szegedy, 2015) `https://github.com/Cadene/pretrained-models.pytorch`
28. `inceptionv3` (Szegedy et al., 2016) `https://github.com/Cadene/pretrained-models.pytorch`
29. `inceptionv4` (Szegedy et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
30. `instagram-48d` (Yalniz et al., 2019) `https://github.com/facebookresearch/semi-supervised-ImageNet1K-models`
31. `mobilenet_v1_tf` (Howard et al., 2017) `https://github.com/tensorflow/models/tree/master/research/slim/`
32. `nasnet_large_tf` (Zoph et al., 2018) `https://github.com/tensorflow/models/tree/master/research/slim/`
33. `nasnet_mobile_tf` (Zoph et al., 2018) `https://github.com/tensorflow/models/tree/master/research/slim/`
34. `nasnetalarge` (Zoph et al., 2018) `https://github.com/Cadene/pretrained-models.pytorch`
35. `nasnetamobile` (Zoph et al., 2018) `https://github.com/Cadene/pretrained-models.pytorch`
36. `pnasnet5large` (Liu et al., 2018) `https://github.com/Cadene/pretrained-models.pytorch`
37. `pnasnet_large_tf` (Liu et al., 2018) `https://github.com/tensorflow/models/tree/master/research/slim/`
38. `pnasnet_mobile_tf` (Liu et al., 2018) `https://github.com/tensorflow/models/tree/master/research/slim/`
39. `polynet` (Zhang et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
40. `resnet101` (He et al., 2016a) `https://github.com/Cadene/pretrained-models.pytorch`
41. `resnet152` (He et al., 2016a) `https://github.com/Cadene/pretrained-models.pytorch`
42. `resnet18` (He et al., 2016a) `https://github.com/Cadene/pretrained-models.pytorch`
43. `resnet34` (He et al., 2016a) `https://github.com/Cadene/pretrained-models.pytorch`

44. `resnet50` (He et al., 2016a) `https://github.com/Cadene/pretrained-models.pytorch`
45. `resnet_v1_101_tf` (He et al., 2016a) `https://github.com/tensorflow/models/tree/master/research/slim/`
46. `resnet_v1_152_tf` (He et al., 2016a) `https://github.com/tensorflow/models/tree/master/research/slim/`
47. `resnet_v1_50_tf` (He et al., 2016a) `https://github.com/tensorflow/models/tree/master/research/slim/`
48. `resnet_v2_101_tf` (He et al., 2016b) `https://github.com/tensorflow/models/tree/master/research/slim/`
49. `resnet_v2_152_tf` (He et al., 2016b) `https://github.com/tensorflow/models/tree/master/research/slim/`
50. `resnet_v2_50_tf` (He et al., 2016b) `https://github.com/tensorflow/models/tree/master/research/slim/`
51. `resnext101_32x4d` (Xie et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
52. `resnext101_64x4d` (Xie et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`
53. `se_resnet101` (Hu et al., 2018) `https://github.com/Cadene/pretrained-models.pytorch`
54. `se_resnet152` (Hu et al., 2018) `https://github.com/Cadene/pretrained-models.pytorch`
55. `se_resnet50` (Hu et al., 2018) `https://github.com/Cadene/pretrained-models.pytorch`
56. `se_resnext101_32x4d` (Hu et al., 2018) `https://github.com/Cadene/pretrained-models.pytorch`
57. `se_resnext50_32x4d` (Hu et al., 2018) `https://github.com/Cadene/pretrained-models.pytorch`
58. `senet154` (Hu et al., 2018) `https://github.com/Cadene/pretrained-models.pytorch`
59. `squeezenet1_0` (Iandola et al., 2016) `https://github.com/Cadene/pretrained-models.pytorch`
60. `squeezenet1_1` (Iandola et al., 2016) `https://github.com/Cadene/pretrained-models.pytorch`
61. `vgg11_bn` (Ioffe & Szegedy, 2015) `https://github.com/Cadene/pretrained-models.pytorch`
62. `vgg11` (Simonyan & Zisserman, 2014) `https://github.com/Cadene/pretrained-models.pytorch`
63. `vgg13_bn` (Ioffe & Szegedy, 2015) `https://github.com/Cadene/pretrained-models.pytorch`
64. `vgg13` (Simonyan & Zisserman, 2014) `https://github.com/Cadene/pretrained-models.pytorch`
65. `vgg16_bn` (Ioffe & Szegedy, 2015) `https://github.com/Cadene/pretrained-models.pytorch`
66. `vgg16` (Simonyan & Zisserman, 2014) `https://github.com/Cadene/pretrained-models.pytorch`
67. `vgg19_bn` (Ioffe & Szegedy, 2015) `https://github.com/Cadene/pretrained-models.pytorch`
68. `vgg19` (Simonyan & Zisserman, 2014) `https://github.com/Cadene/pretrained-models.pytorch`
69. `vgg_16_tf` (Simonyan & Zisserman, 2014) `https://github.com/tensorflow/models/tree/master/research/slim/`
70. `vgg_19_tf` (Simonyan & Zisserman, 2014) `https://github.com/tensorflow/models/tree/master/research/slim/`

71. `xception` (Chollet, 2017) `https://github.com/Cadene/pretrained-models.pytorch`