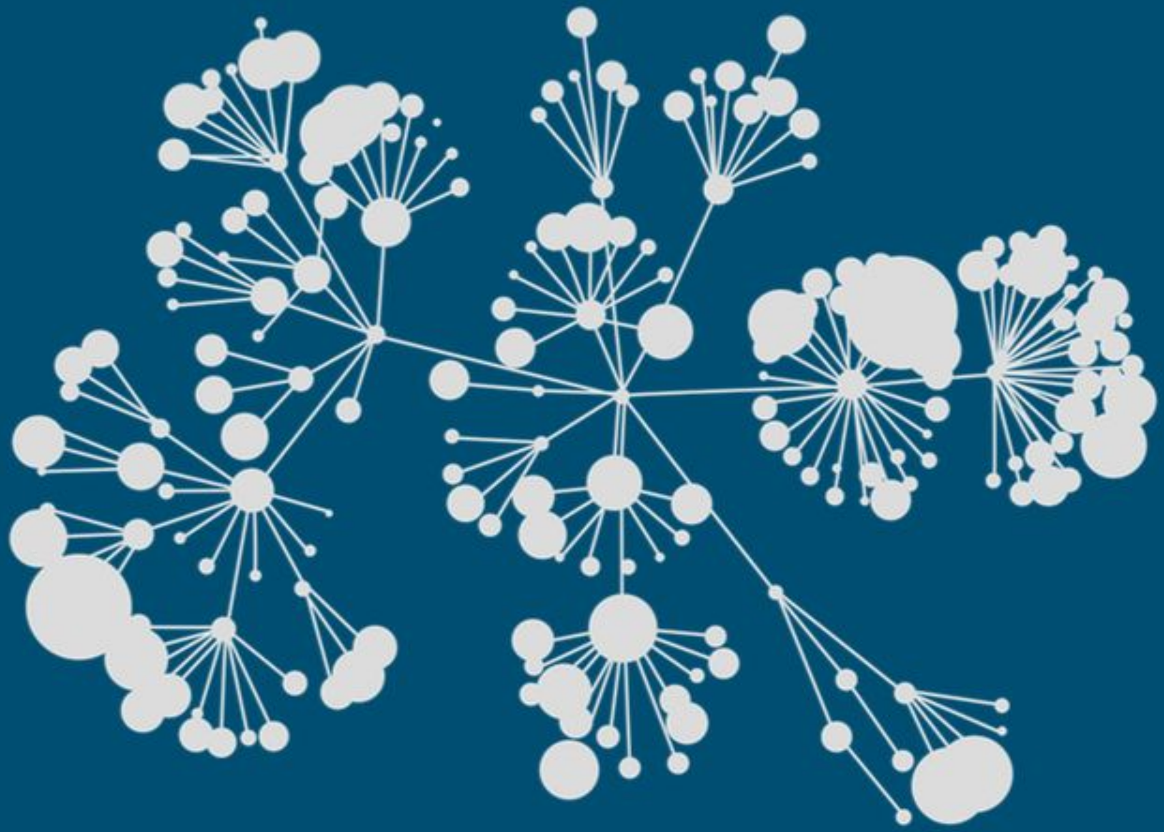
NIPS competition/Kaggle

Inclusive Image Challenge
(3rd place solution)

Team *WorldWideInclusive*

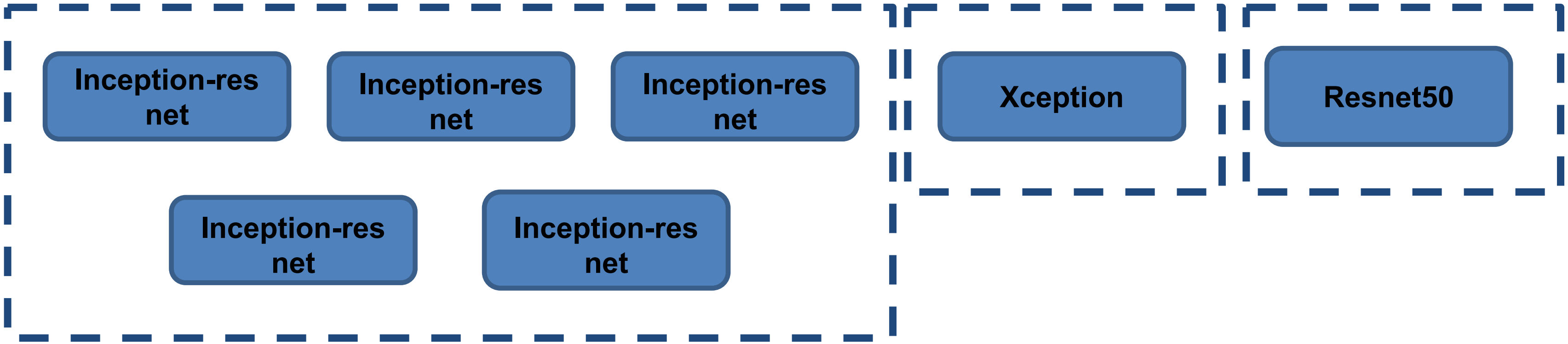


kaggle

Agenda

1. Overall solution architecture
2. Single Model
3. Training details
4. Ensembling & Majority vote
5. Threshold Tuning
6. Important findings
7. Appendix & References

Overall solution - 3 architectures, 7 models



Overall solution - 3 architectures, 7 models

Inception-res net

Inception-res net

Inception-res net

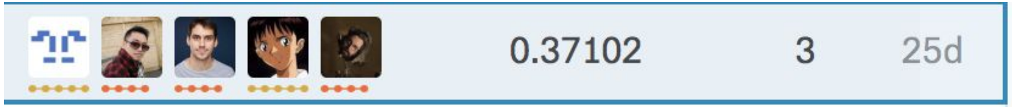
Xception

Resnet50

Inception-res net

Inception-res net

Majority vote (with different threshold settings)



Single model scores

Single inception- resnet	**-**	**0.33735**
Single xception	**-**	**0.32007**
Single resnet	**-**	**0.31501**

Training details

- Both human and machine labels are used for training (with original probabilities)
- Stratified train/dev splits based on class image counts*
  - Train: 1728299 images
  - Valid: 14743 images
- Training with mini batch sizes from 24 - 48, with Adam of learning rate 0.001
- Using validation F2Score (single threshold) for early stopping, and F2Score on tuning labels for monitoring and reference only
- Model that scores higher on validation will also tend to score higher on tuning label set and LB (consistent gap)
- 40,000 images for each epoch, total training takes 200-500 epochs for each model
- We used 299x299 and 336x336 resized images only

Detailed code for stratified splits is in Appendix

Training details - augmentations*

- Two different augmentation libraries:
  - albumentation - https://albumentations.readthedocs.io/en/latest/
  - Keras’ internal ImageDataGenerator (with multi-threaded implemented)

Detailed code for augmentations is in Appendix

Training details - tricks to speed up training

- There are heavy loading and transforming images on CPU side!
- First converting all .jpg images to fixed size (e.g. 299x299) .npz files, and saved to local
- Customized multi-thread version of keras' ImageDataGenerator to load .npz directly!
- Training speed improves by 33%!

Ensembling & Majority Vote

- For each sample, we output labels that are predicted by at least 4 (out of 7) models
- If no labels generated by above approach (around **~19%** of total samples), we will use UNION across all labels of all models
- If still no labels generated (very few, around **~1%**), we will use top 3 most common labels from training
- Ensembling boosts score from **0.33** to **0.37**!
- We also tried using averaged of all predicted probabilities, but the results were no big improvement. (Assuming it is due to different models have very different thresholds)

Ensembling & Majority Vote

- Labeling distribution difference is a more significant factor!
- Average number of labels per image:
    - **Train set - 4.083**
    - **Test set (stage 1) - 2.386**
- To avoid overfitting, we used less aggressive thresholds for our final submission
- In our final submission:
    - 3 models based on (0.1, 0.9) search range
    - 4 models based on (0.01, 0.99) search range

Threshold tuning

- Greedy linear approach based on tuning label set
- For each class - searching in range (0.01, 0.99) - while holding other classes unchanged
- Repeat above step for all classes that have at least *min_n* (e.g. 1) positive sample
- Classes that don't have any positive labels - assign *default_thr* (e.g. 0.99) value
- Repeat above two steps two times - output final threshold array

Important findings - late submissions

Compare to submission with all models using (0.01 - 0.99):

- 48% of total rows are different
- 83% of UNION rows are different (more important)
- 41% of non-UNION rows are different

final_subm_stage_2_7_models_v3_postTestingWithAllThr0.01_0.99.csv 0.36164

9 days ago by Weimin Wang

all 7 models use 0.01 - 0.99

Important findings - late submissions

1 model using (0.1, 0.9)	final_subm_stage_2_7_models_1_model_0.1_0.9.csv 9 days ago by Weimin Wang 1 model with 0.1 - 0.9	0.36584
2 models using (0.1, 0.9)	final_subm_stage_2_7_models_2_models_0.1_0.9.csv 9 days ago by Weimin Wang 2 models at 0.1 - 0.9	0.36871
4 models using (0.1, 0.9)	final_subm_stage_2_7_models_fourModels0.1_0.9.csv 9 days ago by Weimin Wang four models with 0.1 - 0.9 (one more in addition to origin submit)	0.36738
The chosen submission	final_subm_stage_2_7_models.csv 9 days ago by Weimin Wang origin submit	0.37102
0 model using (0.1, 0.9) (all models using 0.01-0.99)	final_subm_stage_2_7_models_v3_postTestingWithAllThr0.01_0.99.csv 9 days ago by Weimin Wang all 7 models use 0.01 - 0.99	0.36164
7 models using (0.1, 0.9)	final_subm_stage_2_7_models_v3_postTestingWithAllThr0.1_0.9.csv 9 days ago by Weimin Wang all 7 models use 0.1 - 0.9	0.36210

Important findings - competition reflection

- Easy to predict labels (objective) : Tree, Plant, Sketch, Guitar, Bicycle, etc.
- Hard to predict labels (abstract) : Tourist attraction, Commercial building, Official, etc.
- FBeta Score not a good metrics:

Beauty, Nature, Plant?, Tree?

Important findings - things that didn't work :(

- Training on a subset (top ~400 labels) of labels- no improvement at all
- Other architectures: NASNet, Mobilenet, DenseNet.
- Training with smaller image sizes (e.g. 224x224)
- Different learning rates other than 0.001
- Average the predicted probabilities for ensembling - not good as majority vote
- Single threshold for all classes (much lower F2 Score)
- Blurred human faces in training

Appendix

- Stratified train/dev splits

```python
if count < 50:
    part = count // 10
elif count < 100:
    part = count // 20
elif count < 1000:
    part = count // 50
elif count < 10000:
    part = count // 100
elif count < 100000:
    part = count // 500
else:
    part = count // 1000

image_files = list(images_for_classes[id].keys())
for img in image_files:
    if img in valid_files:
        part -= 1

np.random.shuffle(image_files)
for img in image_files:
    if img in valid_files:
        continue
    if img in train_files:
        continue
    if part > 0:
        valid_files |= {img}
        part -= 1
    else:
        train_files |= {img}
```

Appendix

- Augmentation for resnet and inception-resnet

```python
def strong_aug(p=.5):
    return Compose([
        # RandomRotate90(),
        HorizontalFlip(),
        # Transpose(),
        OneOf([
            IAAAdditiveGaussianNoise(),
            GaussNoise(),
        ], p=0.2),
        OneOf([
            MotionBlur(p=.2),
            MedianBlur(blur_limit=3, p=.1),
            Blur(blur_limit=3, p=.1),
        ], p=0.2),
        ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.2, rotate_limit=10, p=0.1),
        OneOf([
            OpticalDistortion(p=0.3),
            GridDistortion(p=0.1),
            IAAPiecewiseAffine(p=0.3),
        ], p=0.2),
        OneOf([
            CLAHE(clip_limit=2),
            IAASharpen(),
            IAAEmboss(),
            RandomContrast(),
            RandomBrightness(),
        ], p=0.3),
        HueSaturationValue(p=0.3),
        ToGray(p=0.05),
        JpegCompression(p=0.2, quality_lower=55, quality_upper=99),
        ElasticTransform(p=0.1),
    ], p=p)
```

Appendix

- Augmentation for Xception and inception-resnet

```
gen = ImageDataGenerator(horizontal_flip = True,
                         vertical_flip = True,
                         width_shift_range = 0.1,
                         height_shift_range = 0.1,
                         channel_shift_range=0.1,
                         shear_range = 0.1,
                         zoom_range = 0.1,
                         rotation_range = 10,
                         preprocessing_function=xception_preprocess_input)
```

References

- Kaggle discussion thread: https://www.kaggle.com/c/inclusive-images-challenge/discussion/71433
- https://keras.io/
- https://arxiv.org/abs/1602.07261
- https://arxiv.org/abs/1512.03385
- https://arxiv.org/abs/1610.02357

kaggle™