|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input: application** | **Family** | **Similarity and Difference between families** | **Sub-family: number of modules** | **Similarity and Difference between sub-families** | **Difference within sub-families** |
| Image: generated/discriminate | Compare-gan | - | Model: 15 | * All have only 1 version. * Reference not the same: MODEL one is about hyper parameter tuning. While S3GAN one is aiming at getting similar performance while use less labels. * Dataset used are provided in MODEL ones. Not in S3GAN. * Performance metrics: MODEL ones have MS-SSIM in extra, while S3GAN only have FID and Inception. * MODEL and S3GAN differ in their example code part. * No parameter tuning situation displayed in MODEL ones. * Naming pattern: I don’t know the meaning of “20, 2\_5, 5” in MODEL ones’ names. Not explained. And MODEL ones have dataset, S3GAN don’t have output shape. | * Different hyperparameter settings lead to different modules. * Different tuning lead to different performance. |
| S3gan: 3 | * Cannot find the difference about parameter settings between modules. But they do differ in performances. |
| Image: generate | Biggan | - | Biggan: 3 | * Refer to same paper * **Biggan** have 2 versions, 1 more than **deep** ones. In version 2, they fixed some bugs. | * Different output shape. |
| Biggan-deep: 3 | * Different output shape. |
| Text: embedding | Wiki-words | * Except for U..S..E modules, all the module pages concern such detail: **input, preprocessing, oov**, sentence embeddings. | Wiki-words: 4 | * No changelog. | * **500** and **250** modules have different output shape. * **With-normalization** or **without** modules have different input preprocessing methods: **with-normalization** by removing punctuation and splitting on spaces; **without** ones by splitting on spaces. |
| Nnlm | Nnlm: 28 | * No changelog * Random modules are based on regular nnlm ones. * Random modules use different initialization method comparing with regular ones. * Random ones don’t use input preprocessing normalization. * TF2 ones are implemented in TF2. | * Different language * Different output shape and parameter settings: **dim128** ones will have 3 hidden layers, while **dim50** only have 2. * With or without normalization preprocessing. |
| Random-nnlm: 2 | * Different language. * Output shape different. |
| Tf2-preview-nnlm: 2 | * Different language. * Different output shape and parameter settings: dim128 ones will have 3 hidden layers, while dim50 only have 2. |
| Tf2-preview-gnews-swivel | Tf2-preview-gnews-swivel: 2 | * No changelog. | * One module have “**with-oov**” in its name and another don’t. * They have different expression in vocabulary section: **with-oov**: “ *The vocabulary contains 20,000 tokens with small fraction of the least frequent tokens and embeddings (~2.5%) replaced by hash buckets. Each hash bucket is initialized using the remaining embedding vectors that hash to the same bucket* ”, **the other one**: *Vocabulary contains 20,000 tokens and 1 out of vocabulary bucket for unknown tokens*. |
| universal-sentence-encoder | universal-sentence-encoder: 3 | * Output shape are the same * Xling or single language. * Different reference. Xling ones have one more reference which is related to xling. * No xling ones have 2-3 versions, xling ones don’t have changelog. | * Size: normal > large. Don’t know about the lite one. Probably the lite one is the smallest. * Implementation detail: large one use Transform encoder, normal one use DAN encoder; lite one doesn’t mentioned. * Changelog: v1-v2 are the same. Large one have v3 which fixed some bugs. |
| universal-sentence-encoder-xling: 4 | * Language: 3 of them have 2 different languages, 1 have 8 languages. * Different module have different language combinations(en-fr, en-es, en-de, many) |
| Text: representation | Bert | - | Bert: 6 | * Have changelog, but only for version 1. Not so different from no changelog. | * The text the modules deal with: some are uncased, some are cased, some are multi-cased. I guess this is related to preprocessing method. * The L, H and A respectively means layers, hidden and heads. |
| Video: classification | I3D | - | I3D-kinetics: 2 | * No changelog. * (Use dataset: according to their description, I think the dataset used to train the modules is Kinetics) | * Different classification capability: one module can classify 400 labels while the other one can do 600. |
| Image: classification | imagenet-amoebanet | * Inception v1-v4 (no TF2 one) have the same changelog with resnet v1-v2 * Except for amoebanet one, all of the modules’ output space is 1001 classes. * Inception and resnet modules have same changelog. * TF2 modules have same changelogs. * Amoenabne, pnasnet and nasnet share some similar idea. * PASNet have same changelog with mobilenet-v2. | imagenet-amoebanet: 1 | * No changelog. * Used some of NASNet technique. | * Parameter settings in modules’ name: n means normal cells, f means start with how many convolutional filters. |
| imagenet-nasnet | imagenet-nasnet: 2 | * Have 2 versions. * Version 2 changed some parameters. | * One is large, one is mobile. Mobile one has less normal cells and less conv filters. * Mobile one has smaller output shape. |
| imagenet-pnasnet | imagenet-pnasnet: 1 | * Have 3 versions. * Configured some parameters in version 3, fixed some bug in version 2. | * Only 1 module. It is a large one, largest number of conv filters. |
| imagenet-inception | V1: 1 | * Use different ML models: v1-v4. * There is also a TF2 version of v3. * V1-V4 have the same changelog. They all have 2 versions: various input size, change some parameters, package requirement. * TF2-V3 have 4 changelogs. | * Only have 1 module. |
| V2: 1 | * Only have 1 module. |
| V3: 1 | * Only have 1 module. |
| TF2-preview-v3: 1 | * Only have 1 module. |
| Resnet\_v2 (some people call V4): 1 | * Only have 1 module. |
| imagenet-resnet | V1: 3 | * Use different ML models: v1-v2. * V1-V2 have the same changelog. They all have 2 versions: various input size, change some parameters, package requirement. | * Module details: according to their naming, the main difference is layers of module (50, 101, 152) |
| V2: 3 | * Module details: according to their naming, the main difference is layers of module (50, 101, 152) |
| imagenet-mobilenet | V1: 16 | * Different ML models. * Sub families have different changelogs. V1 have the same changelog with inception modules and resnet modules. V1 quantiops fixed some other bugs. V2 have 3 versions, configured some parameters in version 3, fixed some bug in version 2. TF2-v2 have same changelog with TF2-inception. * V1 and Quantops-V1: the module is somehow quantized by some TF methods. | * Implementation details displayed in their names: a depth multiplier of ?? and an input size of ???x??? pixels |
| Quantops-v1: 16 | * Implementation details displayed in their names: a depth multiplier of ?? and an input size of ???x??? pixels |
| V2: 22 | * Implementation details displayed in their names: a depth multiplier of ?? and an input size of ???x??? pixels |
| TF2-v2: 1 | * Only one version. Depth of multiplier of 1.0 and input size is 224x224 pixels. |
| Image: feature\_vector | (the same with image: classification) | * (nearly the same) * Except for amoebanet one, all of the modules clarify their output vector shape: v1 and quantops: 025-> 256, 050-> 512, 075-> 768, 100-> 1024; v2 and TF2-v2: 35-100 -> 1280, 130-> 1664, 140-> 1792 | (the same with image: classification) | (the same with image: classification) | (the same with image: classification) |
| Image: augmentation | image\_  augmentation | - | image\_  augmentation: 3 | * They only have 1 version. Their changelog only have version 1, so basically have no valuable info. * Regular ones didn’t use published augmentation methods, but NAS one do. What they use is AutoAugment algorithm. | * Different operation to an input image. * I guess the order of augmentation operation is corresponding to the order in their names. |
| image\_  augmentation-nas: 3 | * Use different dataset. |