Deep Learning Checklist

- [ ] Codebase is well-organized
- [ ] Model naming is clear and intuitive
- [ ] Experiment logs are accurate and detailed
  Consider MLFlow, W&B or other similar tools and services.
- [ ] Essential metadata for each model is available
  Dataset version, train script version, and training parameters.
  Consider using a DVC tool.

- [ ] Original data visualization scripts/tools are used
  To ensure accurate interpretation and adequacy of labels.
- [ ] Original data analysis is conducted
  Evaluating characteristics like class count, sample distribution by class, object size distribution for detection, and pixel distribution in masks, among others.

- [ ] Data has been converted to an optimal format
  Consider HDF5 – one of the most convenient formats.
  To reduce volume and disk load, it is advisable to store data in 8-bit if acceptable.
- [ ] Split into Train and Test has been executed as separate sets
  Ideally, Test and Validation should also be distinguished.
- [ ] Data in the databases/sets are randomly shuffled
- [ ] The relationship between the original data and the data in the databases is preserved
- [ ] Metadata is associated with the data
  E.g. attributes in HDF5 store the version of the data generation script, parameters, etc.
- [ ] Developed a script for visualizing data from the database
  Thereby ensuring the correctness of data storage in the database.

- [ ] Quality evaluation metrics are appropriate for the current task
  IoU, Dice Scr., MSE, Recall/Precision, F-Score, Accuracy, ROC/AUC, Confusion Matrix.
- [ ] Standard methodologies for evaluation utilize standard packages
  sklearn.metrics, tf.metrics, ignite.metrics, etc.
- [ ] Evaluation can be conducted separately from the training procedure
- [ ] The quality of a baseline or trivial solution has been evaluated
  In the case of a trivial solution, the evaluation can even be analytical. For example, a random result based on uniform distribution or distribution based on sample analysis or a fixed most probable outcome.

- [ ] Augmentation is computationally efficient
  GPU is used if available.
- [ ] Augmentation correctly accounts for labeling
  Typical problems: points order after flipping, incorrect rotation of a binary mask.
- [ ] Augmentation scripts allow for visual verification of their correctness
- [ ] Augmentation is sufficiently diverse
  Affine transformations (including flip), brightness, contrast, gamma correction, white balance, temperature, noise, blurring, cutout, etc.
  Some tools: ImgAug, DeepMind Augmentation, Albumentations, NVidia DALI.

- [ ] Developed a prediction script for applying the model to an image database
  Relevant for conducting quality evaluation as well.
- [ ] Developed a demo script for applying the model to an individual image
  Can be implemented at a later stage of the project.

- [ ] Visualization of important information during the training process is performed
  E.g. Loss, Train/Test/Val Quality, examples of current results.
  Some visualization packages: Visdom, TensorBoard, TensorBoardX.
- [ ] The training script works with normalized data
  E.g. data is normalized to the range [0, 1] or the mean is subtracted and divided by the variance (with the assessment of statistical metrics).
- [ ] The training script carefully manages IO/disk usage
- [ ] Memory consumption is monitored
  Memory does not "leak". Batch-size is chosen to maximize memory usage.
  Use utilities like htop and nvidia-smi for monitoring.
- [ ] Scripts intended for long-term use support pausing/resuming
  For example, the model state is periodically saved during the training process.
- [ ] Scripts have an adequate list of parameters
  Provide a reasonable amount of command line arguments. Consider Click, Fire, Typer to implement CLI and JSON or YAML to support configuration files. Do not hard-code paths.

- [ ] An adequate amount of computational resources in an appropriate configuration has been allocated
  Care for: the number of servers and GPUs, the topology of GPU interconnections, CPU and GPU performance, the amount of main memory and video memory, etc.
- [ ] Data on computational servers are stored on optimal disks
  There is enough space, IOPS metric is taken into account, prefer local disks over networked disks, prefer SSD over HDD.
- [ ] Backup copies of critically important data are stored in a secure location
  For example, on cloud storage or a dedicated reliable storage server.

- [ ] Standard architectures have been considered/tested
  E.g. ResNet, Inception, MobileNet, EfficientNet, ViT, Swin, UNet, U^{2}Net, PSPNet, MaskRCNN, SSD, Yolo, FasterRCNN, CenterNet, etc.
  Take a look into paperswithcode.com for current SOTA.
- [ ] The network is capable of overfitting on a micro-dataset
- [ ] An analysis of the best and worst predictions of the network is regularly performed
  Preferably on both the training and test datasets.
- [ ] The network architecture and the number of parameters match expectations
  Convenient tools for architecture visualization: NETRON, TensorBoard.

👉 **api4.ai**
👉 **notion**
👉 **pdf**