Blogs:
https://teletype.in/@atmyre/J6qcyBNcPcw (I like the 3rd approach)

Papers:
https://arxiv.org/abs/1912.13458v2
https://arxiv.org/pdf/2302.10174.pdf

Benchmarks:
https://paperswithcode.com/task/fake-image-detection
https://paperswithcode.com/task/deepfake-detection

Repos:
https://github.com/topics/fake-image-detection
https://github.com/agusgun/FakeImageDetector/blob/master/fake-image-detection.ipynb
https://github.com/mjkwon2021/CAT-Net
https://github.com/nogibjj/Detecting-AI-Generated-Fake-Images/tree/main
https://github.com/polimi-ispl/icpr2020dfdc

Datasets:
https://paperswithcode.com/dataset/artifact – Multiple sources
https://www.kaggle.com/datasets/ciplab/real-and-fake-face-detection – Photoshop
https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces – StyleGAN + Flickr
https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images Cifar + SD

What ChatGPT says:

Designing and building a small dataset to differentiate between real and fake face images:
a. Considerations for data collection:
When deciding what data to collect for the dataset, several considerations need to be taken into account. These considerations may include:

Balance: Ensuring a balanced distribution of real and fake images to avoid any bias towards one class.
Agree: less inductive bias

Diversity: Collecting images from various sources, lighting conditions, angles, and ethnicities to create a diverse dataset that is representative of real-world scenarios.
Agree: better generalization

Quality: Ensuring that the collected images are of high quality and have sufficient resolution for accurate analysis and classification.
Obvious

b. Data collection process:
To collect the data, one approach could be a combination of manual collection and utilizing existing datasets. Here's a possible process:

Manual collection: Search for and collect real and fake face images from different sources such as social media, stock photo websites, or online forums. This can involve using web scraping tools or manually downloading images.
Garbage in – garbage out

Existing datasets: Utilize publicly available face datasets that include both real and fake images.
The Jedi way. We will use the balanced subset of the ArtiFact dataset

Examples include the CelebA dataset, which contains a mix of real and fake celebrity face images.
c. Additional labels and sampling method:
In addition to the real/fake labels, it might be beneficial to consider other labels that could enhance the dataset's usefulness. Some potential labels could be:

Lighting conditions: Bright, dim, natural light, artificial light.
Image quality: Blurred, noisy, low resolution, high resolution.
Pose: Frontal view, profile view, tilted head.
This is more about augmentations

To sample a uniform dataset in the i.i.d. (independent and identically distributed) sense, you can randomly select an equal number of images from each label category. For example, if you have three labels (real, fake, blurred), you would randomly sample the same number of images from each category to ensure balance.
Not sure that we can always have these labels

d. API for storing and organizing meta information:
Pandas is a popular data manipulation library in Python, but it is typically used for tabular data. For organizing and storing metadata related to the dataset, a more suitable option would be to use a combination of JSON files and a cloud storage service like Amazon S3. JSON files can store structured information about each image, such as file paths, labels, and additional metadata. These files can be stored in a directory structure, with each file representing an image in the dataset.
There are different ways to store structured data besides JSON: xml, h5, csv

Evaluating classifier performance:
a. High accuracy on the validation set:
While high accuracy on the validation set is promising, it is important to conduct further analysis before drawing final conclusions. Accuracy alone might not provide a comprehensive understanding of the classifier's performance. It's essential to consider other metrics like precision, recall, and F1 score, especially if the dataset is imbalanced.

It is well known that it is better to use Matthews correlation coefficient or Precision-Recall curve or pure Confusion Matrix
https://ieeexplore.ieee.org/document/9440903
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6941312
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7863449
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5721660

b. Additional analysis:
To gain a deeper understanding of the classifier's performance, you can perform the following analyses:

Confusion matrix: Analyze the distribution of true positives, true negatives, false positives, and false negatives to identify specific areas where the classifier might struggle.
Cross-validation: Perform cross-validation to assess the model's performance on multiple folds of the data and ensure that the accuracy is consistent across different subsets of the dataset.
Error analysis: Examine misclassified samples to identify patterns, common sources of errors, or potential weaknesses in the model's predictions.
Yes

Building and testing a simple deep classifier:
We will fine-tune BEiT-3

a. Dataset division:
The dataset can be divided into training and test sets using a standard split, such as an 80:20 ratio. 80% of the dataset can be used for training, while the remaining 20% can be reserved for testing the model's accuracy and generalization.
Yes

b. Data augmentation techniques:
To handle out-of-distribution or unseen images, data augmentation techniques can be applied during training. Some commonly used techniques for face image data augmentation include:

Random rotations: Rotate the images by a certain degree to simulate variations in head pose.
Horizontal flips: Flip the images horizontally to account for mirror images.
Random cropping: Perform random cropping of the images to simulate different face sizes and positions within the frame.
https://github.com/albumentations-team/albumentations Actualy, BEiT has pretty decent built-in ones

Dataset management for millions of images:
a. Faster access on cloud infrastructure like S3:
To achieve faster access to data stored on cloud infrastructure like Amazon S3, you can employ techniques such as parallel processing and caching. Parallel processing can be used to load multiple images concurrently, taking advantage of multi-threading or distributed processing frameworks. Caching can help store frequently accessed data in memory or a faster storage medium, reducing the latency of subsequent access requests.
I believe RAID of several 1 TB SSD would be enough

b. Faster data re-sampling for custom datasets:
For faster data re-sampling to create custom datasets, you can utilize indexing and metadata information. By maintaining indices or metadata that indicate relevant attributes of the images (e.g., labels, attributes), you can quickly filter and select the desired subset of images for creating custom datasets without the need for scanning or iterating through the entire dataset.
Yes

c. Faster data-loader access for faster training:
To accelerate the data-loading process for faster training, you can employ techniques such as prefetching and parallel data loading. Prefetching involves loading data ahead of time into memory or GPU, overlapping the data loading process with model training. Parallel data loading can be achieved by using multiple data loader instances that load and preprocess different batches of data concurrently, reducing the data loading bottleneck.
Yes