CASE STUDY · COMPUTER VISION · IDENTITY AND GRADE DRIFT

Automated QA Gate for AI-Generated Campaign Imagery

Catching identity drift and colour drift in generated fashion photography, before a client sees it. A quality gate that flags good images gets switched off inside a week, so the target was precision, not coverage.

> A Siamese network over frozen ResNet50 features reached F1 0.90 at precision 1.00 on 90 sealed test pairs, against pixel difference at 0.59 and kNN at 0.26. The best result came from changing a number nobody chose: the threshold moved from 0.5 to 0.4821 and recovered all 8 misses.

Taiwo Alabi · MSc Artificial Intelligence, National College of Ireland

Jun 2026 to Aug 2026 · Python · TensorFlow/Keras · ResNet50 (frozen) · scikit-learn · NumPy

01 · The problem

Two distinct faults, so two distinct checks

A fashion brand needs thirty photographs of one model in one outfit, shot from several angles. The images are generated rather than photographed, and they come back looking almost right.

Then you look properly. In shot four the jaw is slightly different. By shot twenty it is a different woman wearing the same jacket. The lighting drifts the same way, warm in one frame and cold in the next, as though the shoot ran over three days instead of three minutes.

No single image looks wrong. The set does not match, and a set that does not match cannot ship. Checking by eye works for thirty images. It does not work for three hundred a day, and it is exactly the kind of repetitive comparison a person gets worse at the longer they do it.

FAULT	WHAT IT LOOKS LIKE	WHERE IT SHOWS UP
Identity drift	The generated person stops being the contracted subject	Between shots taken on different camera angles
Grade drift	The colour and lighting wander off the approved look	Across a multi-image campaign set

The obvious off-the-shelf fix failed first. The `face_recognition` library is built for real photographs, and its distance rule does not transfer to the smoothed skin texture of a generated face. That failure is what made this a project rather than a library call.

02 · Results

The number that matters is precision: 1.00

Measured on a sealed set of 90 image pairs, never touched during training or tuning.

MODEL	PRECISION	RECALL	F1	ROC-AUC
Siamese (proposed)	1.00	0.82	0.90	1.00
Pixel difference	1.00	0.42	0.59	n/a
kNN over same embeddings	0.39	0.20	0.26	n/a

Quick read on the columns:

- **Precision** is how often the gate was right when it said "same person".
- **Recall** is how many of the true same-person pairs it actually caught.
- **F1** blends the two.
- **ROC-AUC** scores how well the model separates the two classes overall. 1.00 is perfect, 0.5 is a coin flip.

The number that matters is precision: 1.00. Every time the gate said "same person", it was right. It never once passed the wrong subject through to a client. Under the cost model for this problem that is the expensive error, and the system made none of it.

> **kNN is the useful comparison.** It reads the identical fingerprints as the proposed model and still scores worst of the three, because it weighs all 128 dimensions equally. Background tint votes as loudly as jaw shape. Same information, no learning, and the score collapses. That gap is the argument for the design.

---

03 · How the system works

Three branches, one gate

Three branches run on the same input. The Siamese network scores identity. The autoencoder scores grade fit. The pixel difference and kNN baselines run on the same test set and lose to both. An image passes only if identity is high AND grade fit is below threshold.

One image comes in. It is prepared once, then scored twice, and it passes only if both checks clear. The two faults are independent: an image can be exactly the right person in exactly the wrong light.

> **The gate fails in the cheaper direction on purpose.** A good image wrongly rejected costs

> one regeneration. A bad image wrongly accepted reaches the client.

---

04 · The identity check

Borrowed features, learned weights

You cannot ask a computer "is this the same woman". So each photograph becomes a list of numbers, a fingerprint, and the question becomes whether two fingerprints sit close together.

Making good fingerprints needs a network that already knows how to look at pictures, and learning that takes about a million photographs. I have 106. So I borrow: ResNet50, pretrained on ImageNet, kept frozen. My 600 pairs cannot improve on a million images, and freezing keeps the remaining job small.

A Siamese network is two identical copies of that frozen network side by side. Photo A down one arm, photo B down the other, through the same weights, so it is one ruler measuring both things. Subtract the two fingerprints, drop the minus signs, and a single trainable layer reads the 128 differences and returns a confidence between 0 and 1.

That final layer is where the design earns its keep. It holds one weight per dimension, so it can learn that jaw shape matters and background tint does not. kNN cannot: it has nothing that learns.

---

05 · The colour check

An autoencoder that only ever sees good work

The problem here is that no bad examples exist. The pipeline never saved its rejects, so supervised classification is not available.

An autoencoder solves this by only ever seeing good work. It is handed an image, made to squash it down to almost nothing, then asked to redraw it from what survived. Each image carries 49,152 numbers and all of it has to pass through a gap 64 numbers wide. Describing a painting in eight words, then having someone repaint it from those eight words, is the same exercise. Memorising is impossible, so only the general character of the image survives, which is exactly the colour and lighting I care about.

Feed it an image with the wrong colours and the redraw comes out badly wrong. The size of that failure is the alarm. It detects bad images without ever having seen one.

INPUT TYPE	RECONSTRUCTION MEAN SQUARED ERROR
Good images	0.00374
Damaged images (60-degree hue shift)	0.00865
Ratio	2.3x

Where the rejection line sits is a business decision, not a technical one:

THRESHOLD	CATCHES	FALSE ALARMS
90th percentile	62%	10%
95th percentile	26%	5%
99th percentile	3%	1%

> **I report the whole curve rather than pick one.** The right answer changes per campaign. Worth stating plainly: 62% is not a solved problem. This is a filter that flags images for a human, not a gate that rejects them unattended.

---

06 · Where it fails

All eight errors are the same error

The 8 same-person pairs the gate missed. Different angle, different lighting, different hairstyle. The honest read is that 90 pairs is too few to claim the gap to the baselines is not luck.

Every single error is the same error. All eight are same-subject pairs the model scored as different, and all eight are back or profile shots, the angles where least of the face is visible. The model was not confused about the face. It could not see the face.

When all your errors share one shape, you know what to fix.

---

07 · The finding I did not expect

The best result came from changing a number nobody chose

The network outputs a score from 0 to 1, and anything above 0.5 was being called "same person". That 0.5 is just the sigmoid default. Tuned on the validation set only, never the test set, the right value is **0.4821**.

That one change recovered all 8 misses and created no new false positives, lifting F1 from 0.90 to 1.00.

THRESHOLD	PRECISION	RECALL	F1	NOTE
0.5 (default)	1.00	0.82	0.90	Sigmoid midpoint, not a decision
0.4821 (tuned)	1.00	1.00	1.00	Recovered all 8 misses, no new false positives

> **The default 0.5 was not a fact, it was a habit.** The best result in the project came from questioning a setting that was never a decision in the first place.

`08 · Honest limitations`

- **The dataset is small.** 106 images, two subjects, 90 test pairs. Two subjects cannot tell me whether this generalises to a third, and that is the ceiling on every claim here.
- **The advantage is not statistically proven.** McNemar's test returned p = 0.19 against pixel difference, which does not clear the usual bar for "definitely not luck". The gap between 0.90 and 0.59 is large, but 90 pairs is too little evidence to rule out chance. That is a shortage of data, not a null result, and I would rather report it than omit it.
- **The off-grade images are damaged by me on purpose.** Real drift is likely subtler than a 60-degree hue shift, so 62% is probably optimistic.
- **The face_recognition baseline failed for a different reason than the proposed method.** Real-photo distances do not transfer to generated faces; that is the same kind of distribution shift the proposed method survives. Counting both as evidence requires acknowledging they failed at different failure modes.

09 · Reproduce it

```
pip install -r requirements.txt
python main.py --phase eval
```

Runs in about a minute and regenerates every metric, confusion matrix and ROC curve. Seeded with `seed=42`, so the numbers are identical on every run. Verified across four consecutive runs while writing this up.

Per-image gate latency is 133 ms on CPU, roughly 450 images a minute, against a person managing a few hundred a day.

Stack: Python · TensorFlow/Keras · ResNet50 (frozen) · scikit-learn · NumPy · Matplotlib

10 · Terms used in this report

Precision
of the images the gate called "same person", how many actually were. This is the number I optimised for: a wrongly-accepted image reaches a client.

Recall
of the images that truly were the same person, how many the gate actually caught.

F1 score
one number blending precision and recall, so a model can't look good by being great at only one.

ROC-AUC
a single score for how well a model separates two classes overall. 1.00 is perfect, 0.5 is a coin flip.

Siamese network
two identical copies of the same model, run side by side on two images, then compared, so it's one ruler measuring both things.

Autoencoder
a model that learns to compress an image down to almost nothing, then rebuild it. Feed it something it's never seen the shape of, and the rebuild comes out visibly wrong. That failure is the alarm.

McNemar's test
a statistical check for whether two models genuinely differ, or just got lucky on this particular set of test images.

Links

Project page: iamtamilore.github.io/p/image-qa

Source: github.com/iamtamilore/campaign-image-qa

Confusion matrix: confusion matrix

Architecture: architecture diagram

Corpus: iamtamilore.github.io/corpus

Home: iamtamilore.github.io