POLITECNICO
MILANO 1863

AN2DL – Second Challenge Report

Backprop. Brothers

Daniele Uras, Filippo Saccomano, Jonny Arbex Marques, Gabriele Carta

Kaggle IDs: danieleuras2, filipposaccomano, jonnyarbexmarques, gabrielecarta28

11094104, 10771424, 10764433, 11097253

December 20, 2025

1 Introduction and Data Quality

This report describes the solution developed by the *Backprop. Brothers* team for the AN2DL Second Image Classification Challenge. The task is a **4-class** classification of high-resolution histology images into *HER2(+), Luminal A, Luminal B*, and *Triple Negative*.

The main challenge is scale. Discriminative cues are local (cells and micro-structures), while images are large and have variable aspect ratios. A naive global resize would blur fine patterns, while full-resolution end-to-end training is expensive. To address this, we adopt a **patch-based MIL pipeline** that processes only informative tissue regions.

The dataset also provides segmentation masks. However, after manual inspection we found that **most masks are noisy or uninformative**, so we did not rely on them for supervision and we instead select tissue directly from the RGB image.

Before training, we applied lightweight automated sanity checks (e.g., unreadable files and near-empty content) to avoid amplifying noise when extracting many patches per slide.

2 Patch Indexing and Embedding Cache

2.1 Image-level Split

We split the training set at **image level** with a stratified 80/20 split. This prevents leakage across train/validation due to patching and keeps class proportions stable.

2.2 Patch Grid and Tissue Filtering

Each image is decomposed into a regular grid of patches with fixed size and stride. To reduce background, we apply a fast **HSV-based tissue filter** and discard low-information patches. If an image yields too many valid patches, we retain only the **top-K** patches, while forcing at least one patch per image. In practice, this produces bags with a bounded size, typically close to K (train mean ≈ 11.98, val mean ≈ 11.93, with max $= 12$).

When patches are close to borders, we use **reflect padding** so that patch size is consistent without introducing sharp artificial edges.

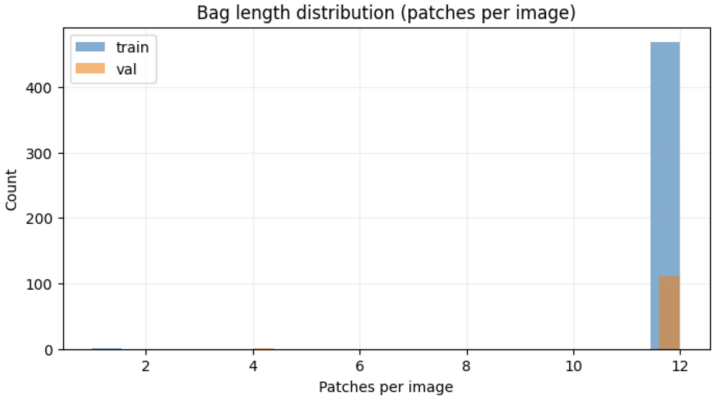

Figure 1: Distribution of the number of patches per image after filtering and top-K selection.

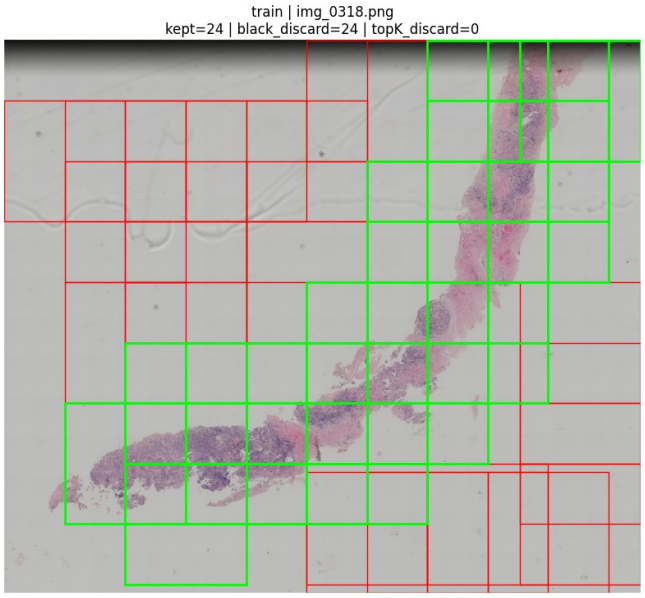

Figure 2: Example of patch selection. Green patches are retained after tissue filtering, red patches are discarded as low-information background.

2.3 Frozen Foundation Embeddings (Cached)

Patches are resized to the foundation model input resolution and passed through a **frozen TF-Keras Path foundation model**. To avoid GPU conflicts with PyTorch training, TensorFlow is forced to run on **CPU**. We cache patch embeddings to disk (`.npz`) and train only on cached features. This reduces training cost and makes multi-seed ensembling feasible.

During embedding extraction, we apply simple stochastic augmentations (flip/rotations) **only on the training split** to improve robustness without changing the validation protocol.

3 MIL Bag Construction

After caching, we build **bags** for Multiple Instance Learning: each image corresponds to a bag of its patch embeddings. Embeddings are sorted by `img_id` and converted into contiguous segments, so bag sampling is efficient.

To reduce overfitting at bag level, we add two training-time perturbations:

- **Patch dropout inside the bag:** we randomly subsample instances, keeping a minimum number of patches.
- **Embedding noise:** we add small Gaussian noise directly in embedding space.

These regularizers are cheap and effective when the backbone is frozen.

Bags have variable length, so we pad to the maximum bag length in the batch and use a boolean **mask** to ignore padding.

4 Model: Gated-Attention Multiple Instance Learning

We use a **Gated MIL** architecture. Given a bag of patch embeddings, the model: (i) projects embeddings to a compact hidden space, (ii) learns attention scores over patches via a gated mechanism, (iii) aggregates a weighted sum of patch features, (iv) predicts the image label from the aggregated representation.

This directly addresses weak supervision: patches are not forced to be individually correct. Instead, the network can focus on a subset of discriminative regions and downweight non-informative tissue.

We also include a lightweight channel re-weighting block (SE-style) to stabilize the pooled representation before classification.

```
=== Gated MIL Head (Quad-class) Architecture ===
GatedMIL(
  (patch_proj): Sequential(
    (0): LayerNorm((384,), eps=1e-05, elementwise_affine=True)
    (1): Linear(in_features=384, out_features=160, bias=True)
    (2): GELU(approximate='none')
    (3): Dropout(p=0.4, inplace=False)
    (4): LayerNorm((160,), eps=1e-05, elementwise_affine=True)
  )
  (se_block): SEBlock(
    (squeeze): AdaptiveAvgPool1d(output_size=1)
    (excitation): Sequential(
      (0): Linear(in_features=160, out_features=20, bias=False)
      (1): ReLU(inplace=True)
      (2): Linear(in_features=20, out_features=160, bias=False)
      (3): Sigmoid()
    )
  )
  (attn): GatedAttention(
    (V): Linear(in_features=160, out_features=96, bias=True)
    (U): Linear(in_features=160, out_features=96, bias=True)
    (w): Linear(in_features=96, out_features=1, bias=True)
  )
  (bag_drop): Dropout(p=0.5, inplace=False)
  (cls): Linear(in_features=160, out_features=4, bias=True)
)
```

Figure 3: Gated-attention MIL architecture. Patch embeddings are weighted through attention, pooled, and passed to the image-level classifier.

5 Training Strategy and Overfitting Control

Training is performed in PyTorch on image-level bags. We monitor **macro-F1** on the validation split, since it is robust to class imbalance and highlights per-class performance.

To improve robustness and limit overfitting, we combine:

- **Frozen backbone:** only the MIL head is trained, reducing capacity and training instability.
- **Bag regularization:** patch dropout and embedding noise applied during training.
- **Loss stabilization:** label smoothing and **R-Drop** consistency regularization.

We also track the **overfit gap** (train macro-F1 minus validation macro-F1) across epochs. Across multiple seeds, the mean validation macro-F1 peaks at ≈ 0.5026 around epoch ≈ 42, after which training performance keeps increasing while validation saturates.

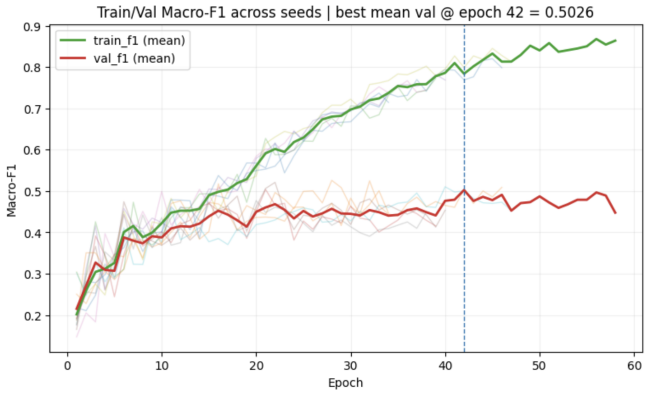

Figure 4: Training and validation macro-F1 across epochs, averaged over multiple random seeds.

6 Ensemble Inference and Submission

To reduce variance, we train multiple runs with different random seeds and build a **multi-seed ensemble**. At inference, we average image-level probabilities across ensemble members and predict the class with maximum averaged probability.

Test-time processing follows the same steps: patch extraction and filtering, frozen embedding extraction, bag construction with padding masks, and MIL inference. Finally, we generate `submission.csv` by mapping predicted ids back to label strings.

7 Patch Importance (Attention-Based)

Instead of pixel-level Grad-CAM, this pipeline provides **attention weights** over patches by design. We use these weights as a qualitative check: high-attention patches should correspond to informative tissue regions, while background should receive low attention.

8 Limitations

Supervision is only available at image level. Even with MIL, performance can be limited when discriminative evidence is sparse or when slides contain large irrelevant areas. In addition, scanner artifacts can still influence the model if they correlate with specific classes.

9 Results

We evaluate performance at image level using macro-F1 as the primary metric. On the validation split, the ensemble achieves:

- **Macro-F1 (VAL, ensemble):** ≈ 0.5192
- **Accuracy (VAL, ensemble):** ≈ 0.5268

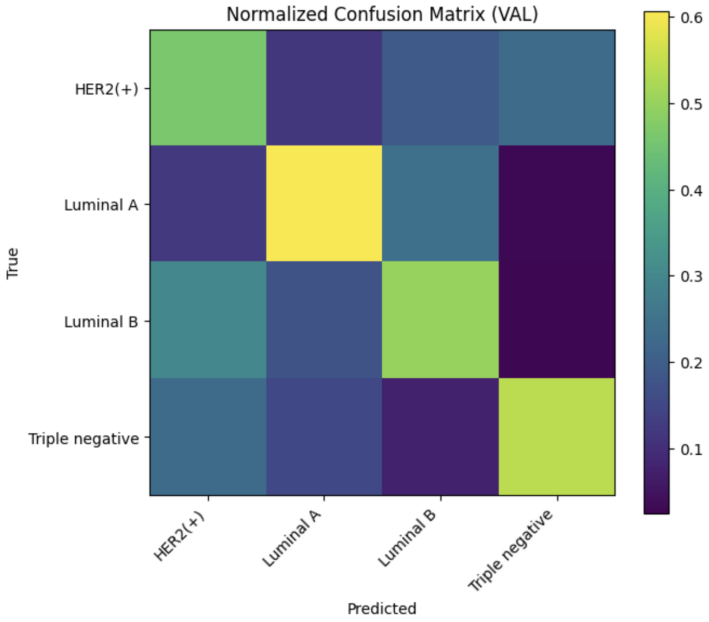

Figure 5: Confusion matrix on the validation set for the gated-attention MIL ensemble.

10 Conclusion

We presented a patch-based solution that caches frozen foundation embeddings and trains a gated-attention MIL classifier on image-level bags. The design is compute-efficient and targets weak supervision by learning which patches matter. Ensembling across seeds further improves stability and final predictions.

A Additional Experimental Results

This appendix reports additional experiments performed to assess the impact of (i) different classification heads on top of the same frozen backbone, and (ii) a simpler baseline trained as a single model with weight averaging. All results are reported at image level using macro-F1 as the primary metric.

A.1 Model B: Path Foundation with Alternative Classification Heads

This set of experiments evaluates the effect of the **classification head design** while keeping the rest of the pipeline unchanged.

Backbone. The Google Path Foundation model is used as a frozen feature extractor to generate patch embeddings, identical to the main model described in the report.

Experimental Setup. All variants share the same data split, patch extraction strategy, and training protocol. Only the classification head is modified.

The following heads are evaluated:

- **Simple MLP Head**: a shallow multi-layer perceptron operating on aggregated patch embeddings.
- **Improved MLP Head**: a more stable variant with Layer Normalization, GELU activations, and dropout.
- **Gated-Attention MIL Head**: a gated attention mechanism that learns patch-level importance weights before aggregation.

Training. All heads are trained using the same optimizer and loss configuration. No ensembling or test-time augmentation is applied in this comparison, so that differences can be attributed solely to architectural choices in the head.

Results. The gated-attention MIL head consistently outperforms simpler aggregation strategies, highlighting the importance of explicitly modeling patch relevance under weak supervision. MLP-based heads show competitive performance but suffer from higher sensitivity to noisy or non-informative patches.

	precision	recall	f1-score	support
HER2(+)	0.3448	0.3846	0.3636	26
Luminal A	0.6071	0.5152	0.5574	33
Luminal B	0.5294	0.4500	0.4865	40
Triple negative	0.3333	0.5385	0.4118	13
accuracy			0.4643	112
macro avg	0.4537	0.4721	0.4548	112
weighted avg	0.4867	0.4643	0.4702	112

...

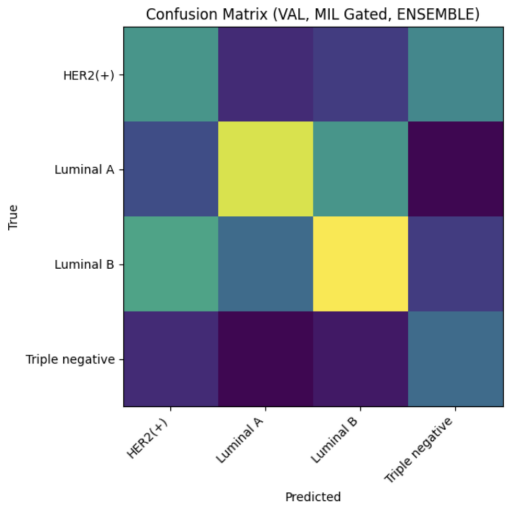

Figure 6: Confusion matrix for Model B, comparing different classification heads trained on the same frozen backbone.

A.2 Model C: Baseline Single Model with Stochastic Weight Averaging

This experiment provides a **baseline reference** by removing architectural complexity and ensembling effects.

Backbone. The same frozen Path Foundation model used in the main pipeline.

Classification Head. A single classification head is trained without architectural variations. No multi-head comparison and no ensemble are used.

Training Setup.

- Single training run (no multi-seed ensemble)
- Cross-Entropy loss with label smoothing
- Optimizer: AdamW with cosine annealing
- **Stochastic Weight Averaging (SWA)** applied during the final phase of training

Results. The SWA baseline improves stability compared to a standard single checkpoint, but remains clearly below the performance of the main multi-seed gated-attention MIL model. This confirms that both architectural expressiveness and ensembling contribute significantly to the final performance.

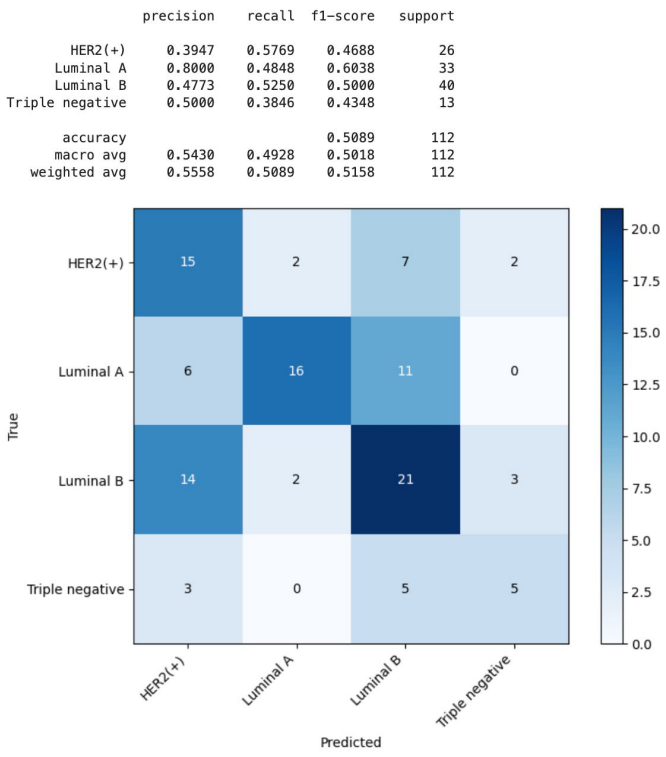

Figure 7: Confusion matrix and classification report for Model C single model baseline with SWA).

A.3 Model D: Path Foundation + ImprovedHead (3-seed Ensemble)

Backbone. Google Path Foundation model used as a frozen patch-embedding extractor.

Classification Head. We replaced the previous larger head with a smaller and more stable MLP (`ImprovedHead`). The head applies LayerNorm before linear layers to stabilize embedding scale, followed by GELU activations and dropout. The final classifier is a LayerNorm + Linear layer.

Ensemble. We trained the same configuration with **3 different random seeds** and performed inference by averaging image-level probabilities across models.

Training Setup.

- Data augmentation: **disabled** (same patch set)
- Loss function: **Focal Loss** (no additional class weights)
- Optimizer: **Lion**
- Patch aggregation: mean probability over patches

Results. On the validation set (112 images, average ∼24 patches/image), the ensemble achieves an accuracy of ≈ **0.455** and macro-F1 of ≈ **0.447**. The confusion matrix in Figure 8 highlights persistent confusion between Luminal A and Luminal B, while Triple Negative remains the hardest class in terms of recall.

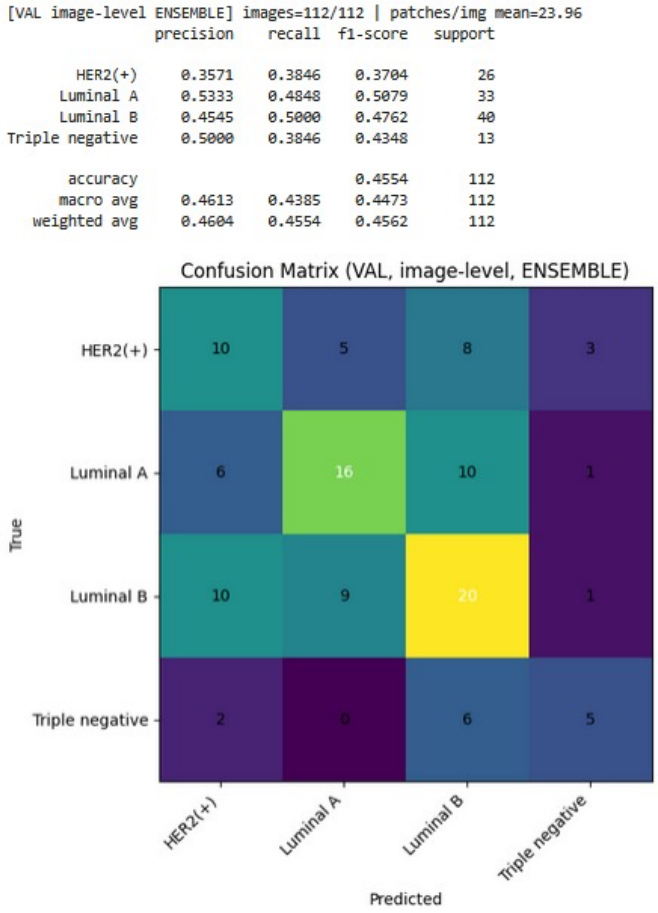

Figure 8: Confusion matrix and classification report for Model D (Path Foundation + ImprovedHead, 3-seed ensemble, focal loss, Lion).

A.4 Model E: EfficientNet-B7 with Test-Time Augmentation

Backbone. EfficientNet-B7 trained end-to-end for image-level classification. The model processes resized full images, without explicit patch decomposition.

Training Setup.

- Loss function: Cross-Entropy with **class weights** and **label smoothing**
- Optimizer: standard SGD/Adam-based training (EfficientNet setup)
- Data augmentation: enabled at training time
- Test-Time Augmentation (TTA): enabled at inference

Inference with TTA. At inference, multiple augmented views of each image are evaluated and the final prediction is obtained by averaging the resulting probabilities.

Results. On the validation set (92 images), this configuration achieves an accuracy of approximately **0.50** and a macro-F1 of approximately **0.48**. As shown in Figure 9, the confusion matrix highlights similar failure modes to the Path-based models, with strong confusion between Luminal A and Luminal B and limited recall for the Triple Negative class. Despite the use of TTA and a large backbone, performance remains comparable to patch-based approaches.

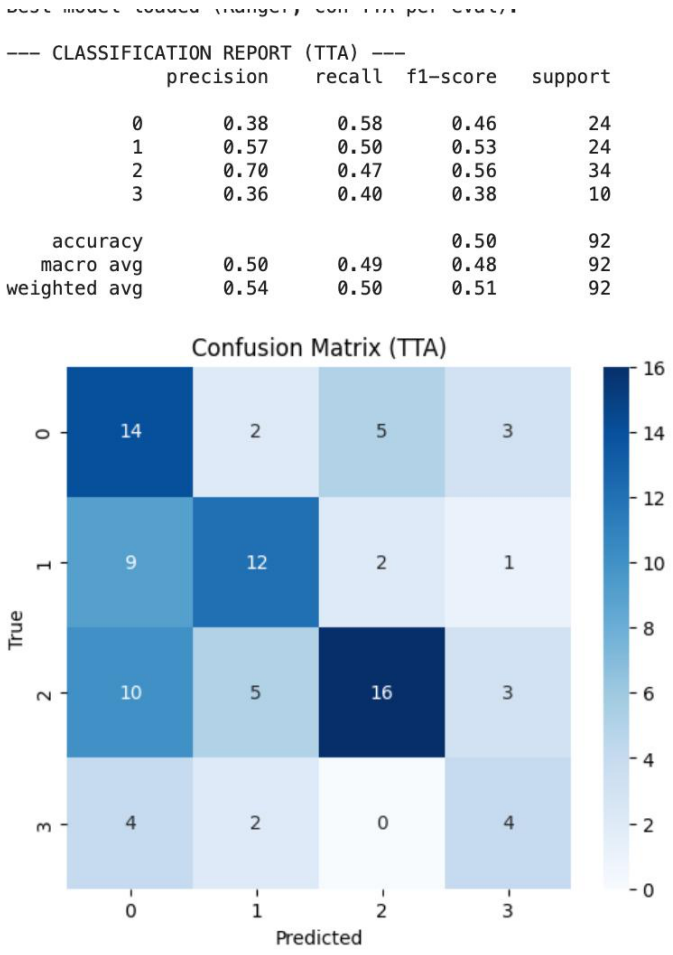

Figure 9: Confusion matrix and classification report for Model E (EfficientNet-B7, class weights, label smoothing, TTA).