National Yang Ming Chiao Tung University
Department of Civil Engineering

Dispersion-Spectra Inversion in Subsurface Vs Profiling using Artificial Intelligence

Comparing different deep learning models for dispersion-spectra inversion

Presenter: CHANTHRABOUTH-LIEBBE Simon (intern)
Supervisor: Prof. CHIH-PING Lin and Dr. TRAN Kinh

August 2026

Summary

1 Context of the project

2 FVS branch - Model performance comparison

3 Geometry branch - Model performance comparison

4 Conclusion

5 Future work

CONTEXT

Why look under our feet?

The goal: know the layers beneath a road or a site without drilling a single hole.

Economic

Roads, runways and foundations are inspected by **coring**: slow, destructive, and only a few points per site.

Societal

Weak or thinning pavement layers are found before they become potholes, sinkholes or closures.

Human

No excavation means no traffic closure, no digging crew exposed to live lanes, and no damage to what is already built.

The method: Multichannel Analysis of Surface Waves (MASW)

Hit the ground with a hammer, record the waves with a line of sensors, and read the subsurface from how those waves travel. Non-destructive technique, and an inverse problem. There are 2 ways to solve this.

CONTEXT

Two ways to solve the inversion

Classical inversion

Hours or days

The computer tries thousands of candidate profiles and keeps the one that best reproduces the measurement **(Monte-Carlo, genetic algorithms, simulated annealing)**. Runtime explodes with the number of layers and the precision required.

Trained model (AI)

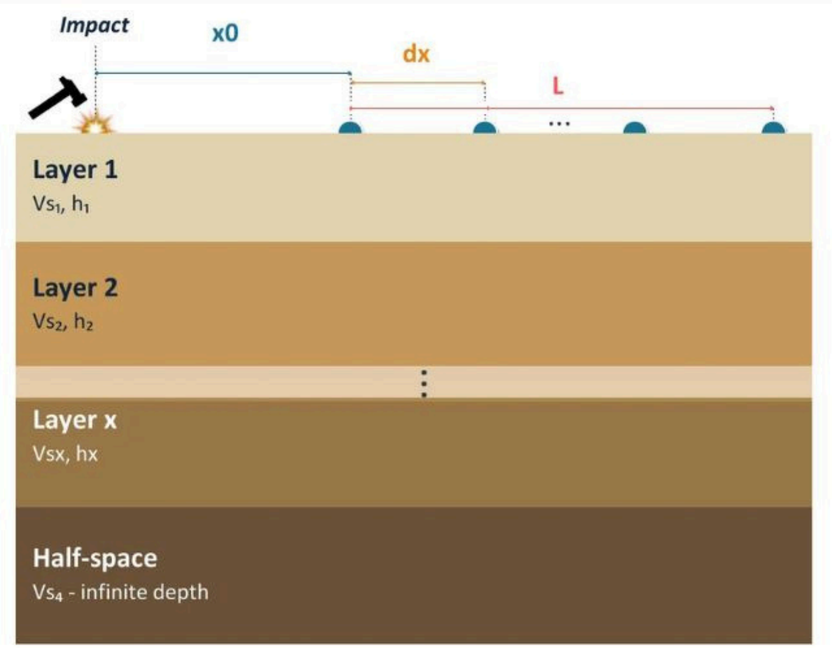
A few seconds

The cost moves to training, which happens once. Thousands of simulated examples teach the network the spectrum → profile mapping; afterwards a single forward pass proposes a Vs profile.

This project focuses on building the AI model that performs the inversion.

How the inversion problem works

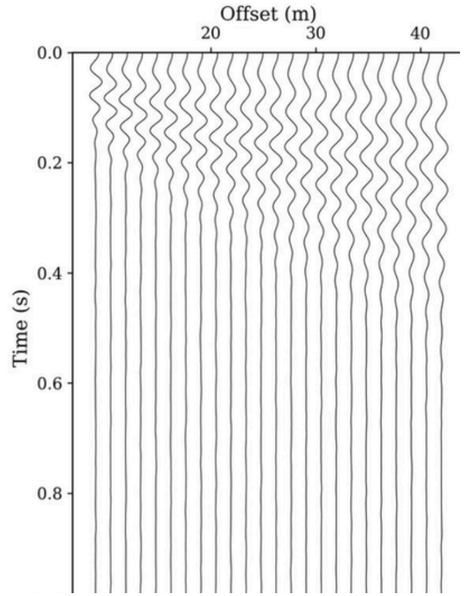
1 Hit and record



A hammer blow sends surface waves through the ground; CH sensors on a line record them.

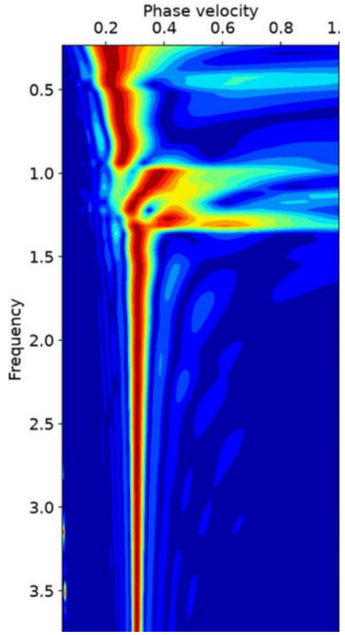
x0: Offset
dx: Distance between receivers
CH: Number of receivers

2 Waveforms



Each sensor records a time signal. Waves arrive later and more spread out with distance.

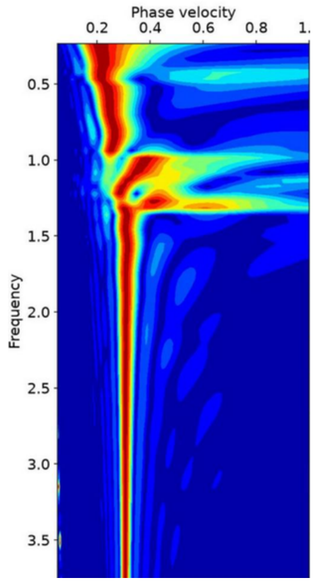
3 Dispersion spectrum



The signals are transformed into one image: **Full Velocity Spectrum** -> Model's input.

How the inversion problem works

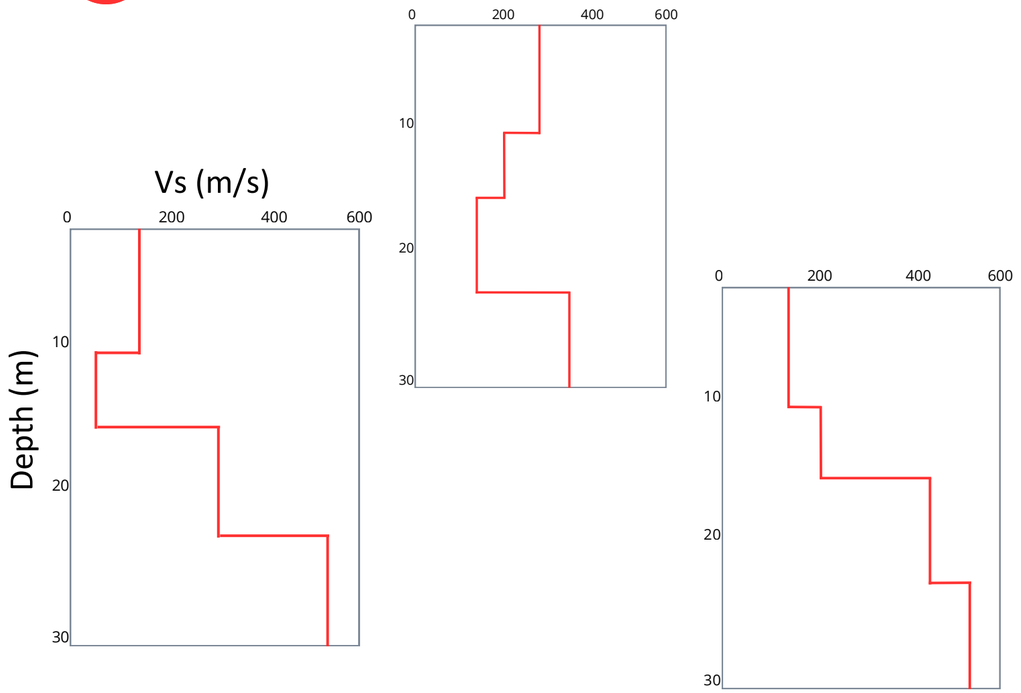
3 Dispersion spectrum



The signals are transformed into one image: speed against frequency. This is the model's input.

≈

4 Vs Profile (shear-wave velocity)



There is no single obvious answer!

Cho et al. (2024)

We need a model that picks the most plausible profile out of that whole space, which is exactly what a network trained on many examples learns to do.

CONTEXT

Training structure example

300,000 input/output pairs were generated in MATLAB. Each input is a **3 × 76 × 191 array** plus **3 geometry numbers**.

INPUT

3 geometry numbers

Parameters	Range
x0' : Normalized Near offset	0 - 1
Ch : Number of receivers	24 or 48
dx : Receiver spacing	1, 1.5 or 2 m

L = (Ch - 1) × dx

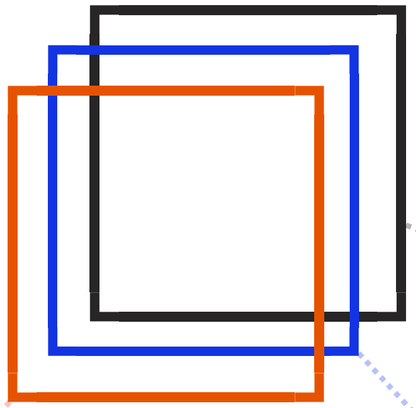
The array length (meter)

x0' is stored as $\frac{x0}{L}$

x0 : [2, 20] m, step of 1 m

.mat file

3 Sheets



Amplitude: 0 to 1

the only real measurement

Normalized value (f'): 0.115 to 7.520

=> Frequency (f): 5 to 80 Hz

Depends on the parameters

$$f' = \frac{f \times L}{Vsmax}$$

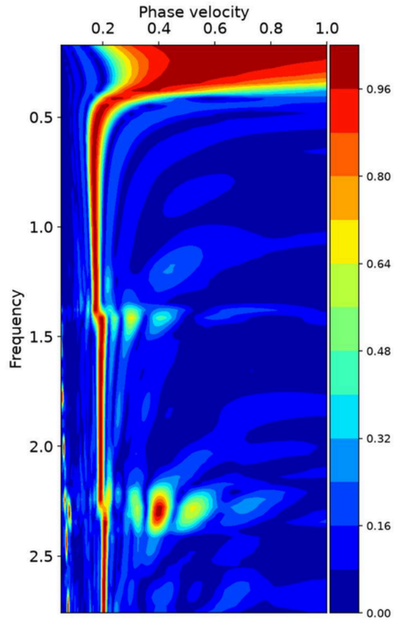
Normalized value (Vs'): 0.05 to 1

=> Phase Velocity: 50 to 1000 m/s (Vs_{Max}= 1000)

same in every file

$$Vs' = \frac{Vs}{Vsmax}$$

FVS image



Training structure example

300,000 input/output pairs were generated in MATLAB. Each input is a **3 × 76 × 191 array** plus **3 geometry numbers**.

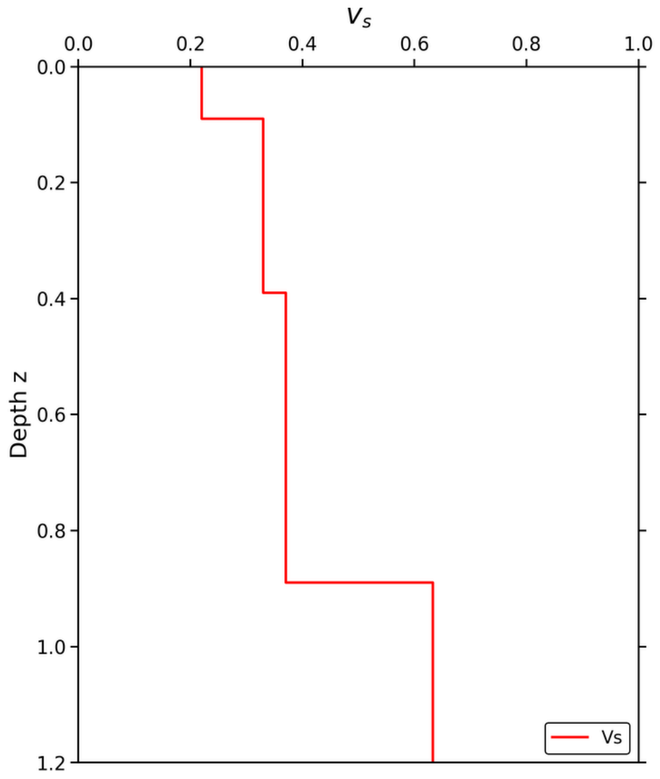
OUTPUT (.csv)

	C1
1	0.12128
2	0.12128
3	0.12128
4	0.12128
5	0.12128
6	0.24048
7	0.24048
8	0.24048
9	0.24048
10	0.24048
11	0.24048
12	0.24048
13	0.24048

101 values = the Vs profile

what the model must predict

Vs profile



Cho et al. (2024)

ModelCNN_fvs (Baseline): Dr. Kinh's starting point

Two branches - one reads the spectrum, one reads the geometry - merged before predicting the profile.

Amplitude matrix size : 76 × 191 = **14,516**

Spectrum image (fvs)
14 516 inputs

Pretrained CNN backbone **(ResNet-50)**
→ **1000 Output**

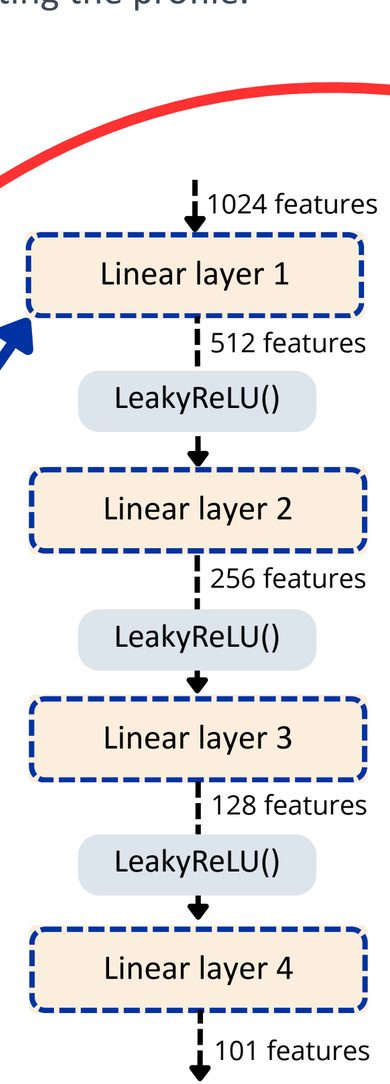
Array geometry (x0')
14 516 inputs

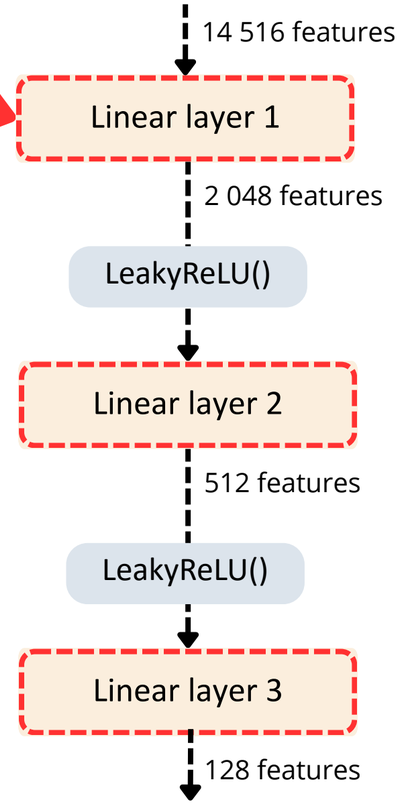
Dense network
x0 tiled, then flattened → **128 Output**

Fusion: the two feature vectors are concatenated
1000 + 128 input → 1024 Output

Prediction head dense layers + LeakyReLU()

Output: Vs profile (csv file) - 101 features





Comparing 6 models

Model	Family	Backbone output	Total params	Idea tested
Baseline CNN	Residual CNN	1000	58.3 M	Dr. Kinh's starting point, **BASELINE**
ResNet-50 (revised)	Residual CNN	2048	57.3 M	A cleaned-up version of the baseline
ResNet-34	Residual CNN	512	53.5 M	Fewer layers, to see if the results are similar
DenseNet-121	Dense CNN	1024	39.7 M	New family, dense connections
EfficientNet-B0	Compact CNN	1280	37.0 M	Best size / accuracy trade-off
Swin Transformer	Transformer	768	60.0 M	Attention, without convolution

All 6 keep exactly the same structure (spectrum branch + geometry branch + prediction head). Only the backbone changes: its first layer is replaced to accept our 3-channel image, everything else keeps its pretrained weights.

The common experimental protocol (without noise)

Hyperparameter	Value
Training/validation split	**80% / 20%**
Data size	**90,000** out of 300,000
Batch size	**8**
Loss function	**RMSE Loss** (Root Mean Squared Error)
Seed (fixes identical starting conditions for a fair cross-architecture comparison)	**42**
Number of epochs	**200**
Early stopping	**45**
Optimizer	**Adam**
Learning rate	**0.0001**
Scheduler (after 20 epochs)	**0.5**

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y'_i - y_i)^2}$$

N: batch size × output size = 808
yi: Vs value at position i
y'i: Vs predicted value at position i

After 45 epochs without improvement, training stops

Every 20 epochs, the learning rate is halved

The best validation epoch is automatically saved.

ResNet-50 (revised)

Residual CNN

2048

backbone output size

57.3 M

total trainable parameters

TECHNICAL PARTICULARITY

- Native head kept (nn.Identity): a raw 2048-feature vector instead of passing through the 1000-class ImageNet layer.
- A LeakyReLU is added right after fusion, fixing two consecutive linear layers with no non-linearity between them.

KEY CONCEPT

Residual (“skip”) connections let information bypass layers, which keeps the signal from fading (vanishing gradient) and makes networks beyond 50 layers trainable.

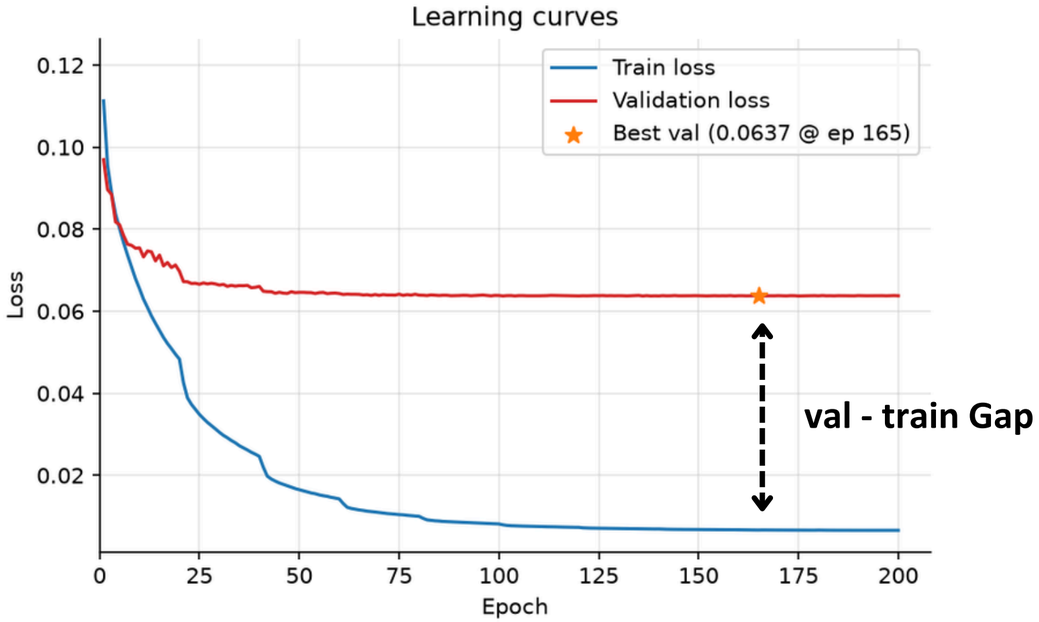
Test objective: *Does a cleaner transfer-learning setup improve the results?*

$$y = F\left(x, \{W_i\}\right) + x$$

x: block input
F(x,{Wi}): block transformation
y: block output

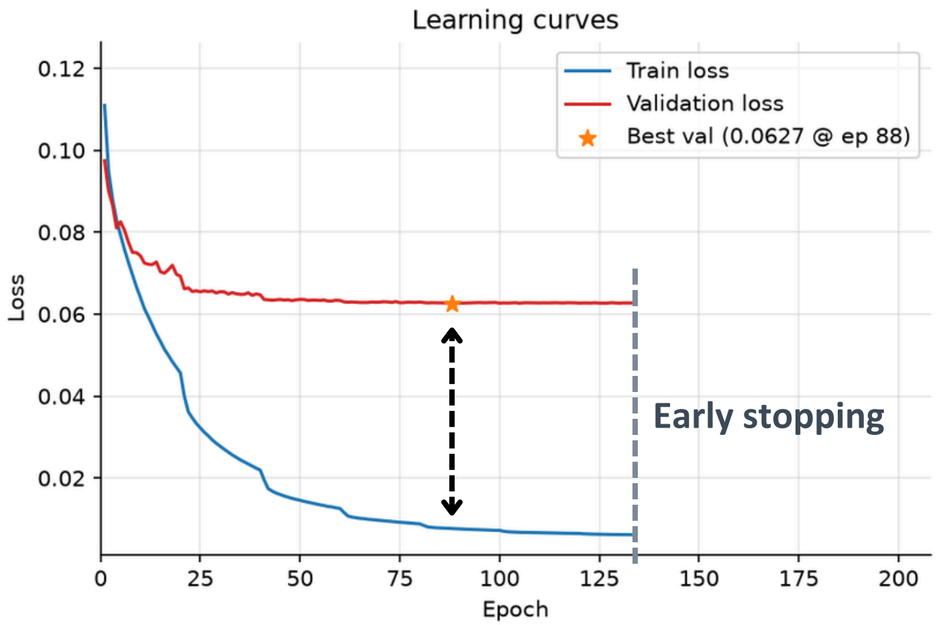
ResNet-50 (revised) vs. Baseline CNN

Baseline CNN



Validation-train gap at best epoch: +0.0570
val/train ratio = 9.57

ResNet-50 (revised)



Validation-train gap at best epoch: +0.0549
val/train ratio = 8.17
=> Both models overfit similarly, the cleanup did not reduce the gap

ResNet-34

Residual CNN

512

backbone output size

53.5 M

total trainable parameters

TECHNICAL PARTICULARITY

- Same family as ResNet-50 but shallower (34 vs 50 layers), with a backbone output four times smaller (512 vs 2048).
- Slightly less capacity for a slightly lower compute cost per epoch.

KEY CONCEPT

Same residual principle, shallower architecture: fewer parameters to learn and faster epochs.

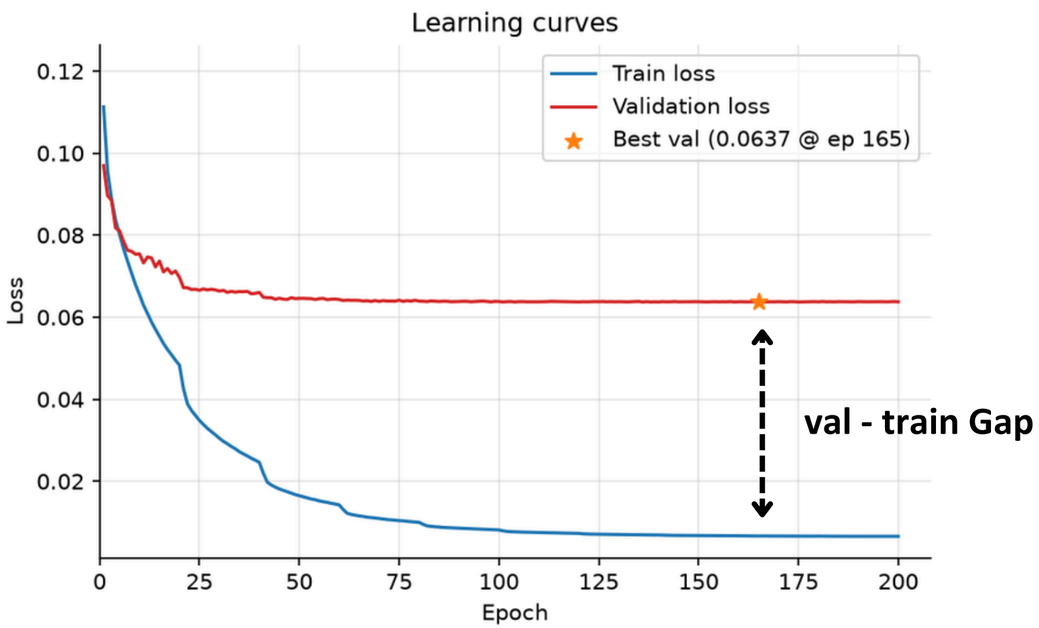
Test objective: Can a lighter model represent the spectrum as well as ResNet-50?

$$y = F\left(x, \{W_i\}\right) + x$$

x: block input
F(x,{Wi}): block transformation
y: block output

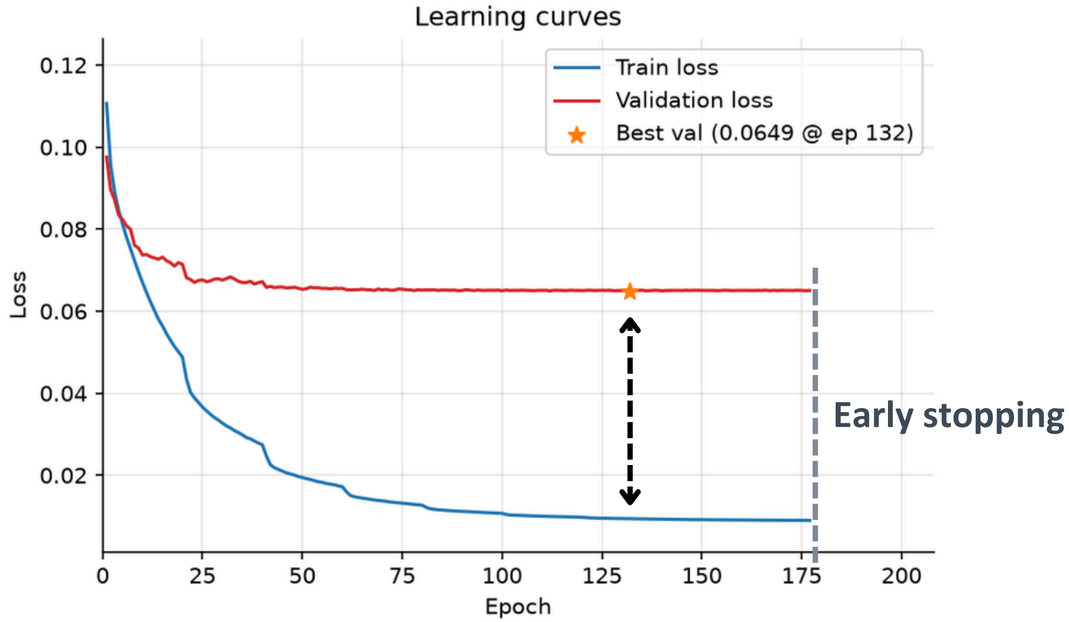
ResNet-34 vs. Baseline CNN

Baseline CNN



Validation-train gap at best epoch: +0.0570
val/train ratio = 9.57

ResNet-34



Validation-train gap at best epoch: +0.0555
val/train ratio = 6.89
=> ResNet-34 overfits less but scores slightly worse

DenseNet-121

Dense CNN

1024

backbone output size

39.7 M

total trainable parameters

TECHNICAL PARTICULARITY

- A different family, not a ResNet variant: every layer receives the concatenation of ALL previous layers, not just the last one.
- The result is a very deep but compact backbone (~8 M backbone parameters).

KEY CONCEPT

Dense connections give each layer a summary of everything already extracted, which cuts redundancy and the parameters needed.

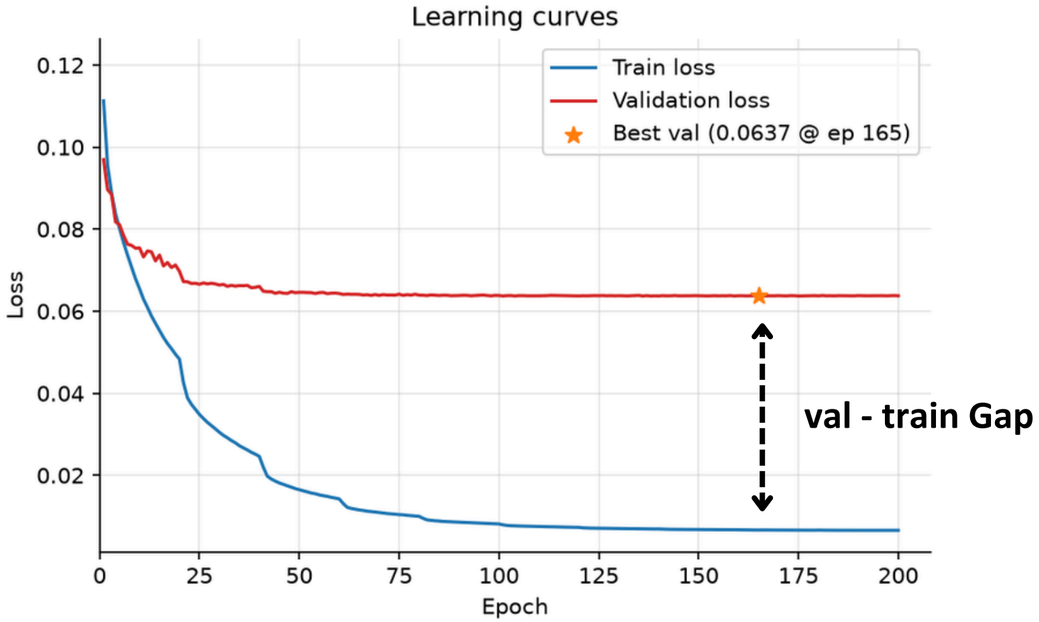
Test objective: Can a more parameter-efficient architecture match or beat the ResNets?

$$x_l = H_l\left([x_0, x_1, \ldots, x_{l-1}]\right)$$

xℓ: output of block ℓ
[x0,…,xℓ−1]: concatenation of all previous features
Hℓ: composite transformation function (BatchNorm → ReLU → Conv)

DenseNet-121 vs. Baseline CNN

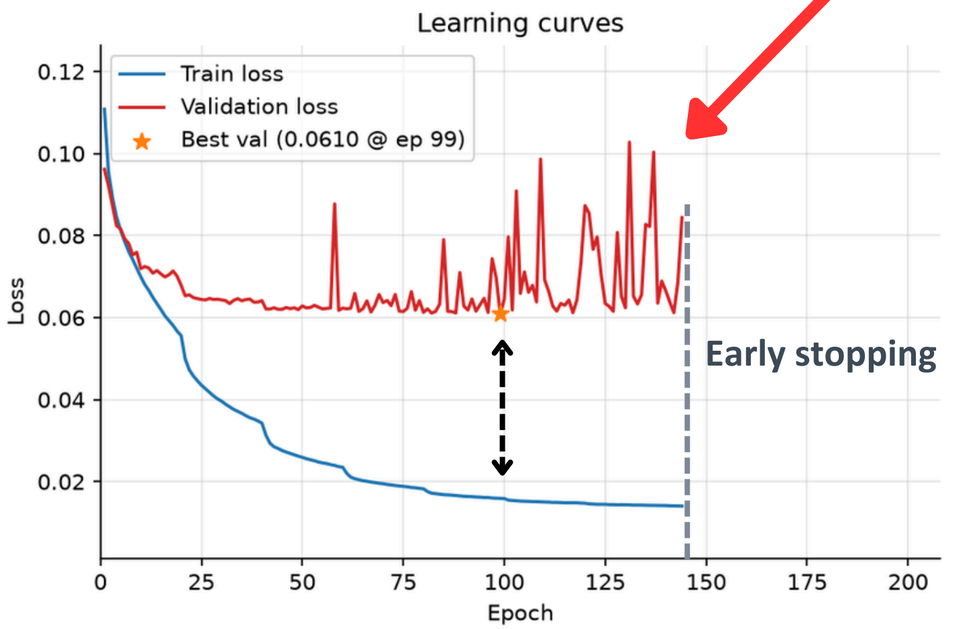
Baseline CNN



Validation-train gap at best epoch: +0.0570
val/train ratio = 9.57

DenseNet-121

Dense connections make BatchNorm more sensitive to the learning rate



Validation-train gap at best epoch: +0.0450
val/train ratio = 3.83
=> DenseNet-121 has the best ratio, but is unstable

EfficientNet-B0

Compact CNN

1280

backbone output size

37 M

total trainable parameters

TECHNICAL PARTICULARITY

- Depthwise-separable convolutions (cheaper to compute) combined with per-channel weighting (squeeze-and-excitation).
- The most compact backbone tested: ~5.3 M

KEY CONCEPT

Squeeze-and-excitation learns how much weight each channel deserves, attention applied to channels rather than pixels.

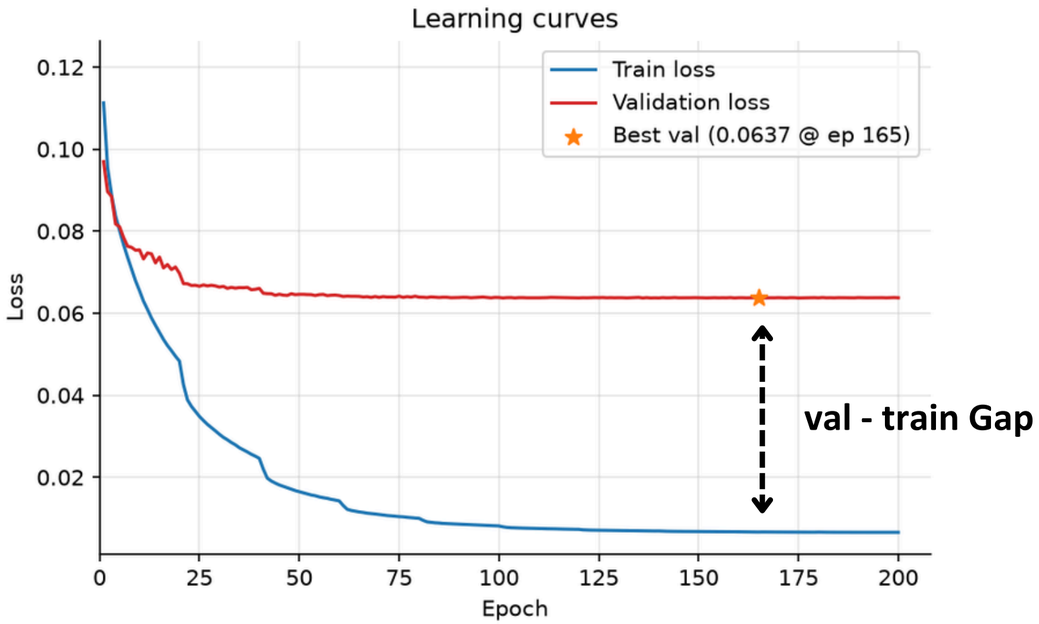
Test objective: Can this architecture match or beat the ResNets?

$$\left.\begin{aligned} d &= \alpha^{\phi} \\ w &= \beta^{\phi} \\ r &= \gamma^{\phi} \end{aligned}\right\} \quad \begin{aligned} &\alpha \times \beta^2 \times \gamma^2 \approx 2 \\ &\alpha, \beta, \gamma \geqslant 1 \end{aligned}$$

d: depth (number of layers)
w: width (number of channels)
r: resolution (image height/width)
ϕ: single user-chosen coefficient controlling d, w, r together
α, β, γ: fixed constants found once by grid search

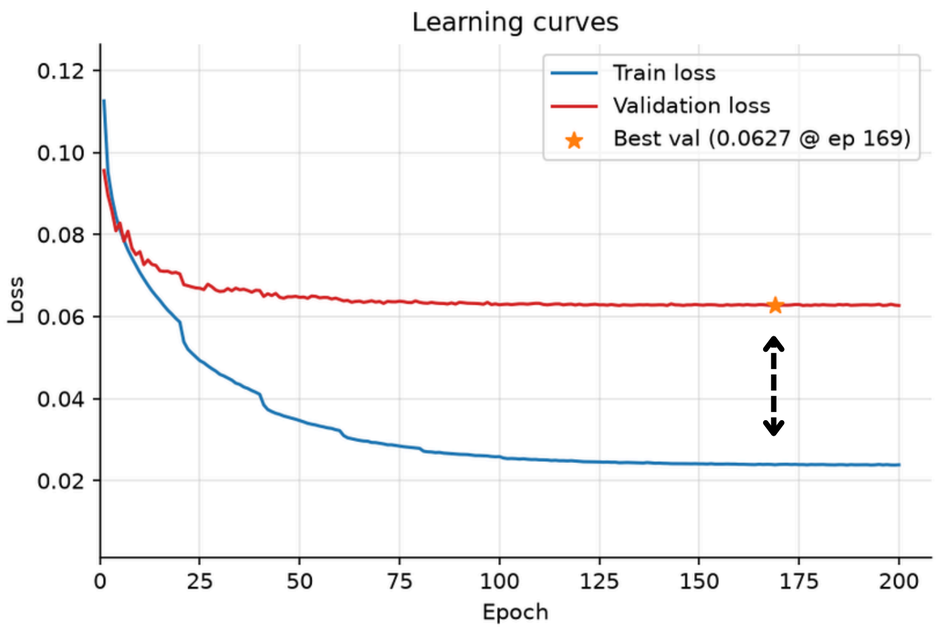
EfficientNet-B0 vs. Baseline CNN

Baseline CNN



Validation-train gap at best epoch: +0.0570
val/train ratio = 9.57

EfficientNet-B0



Validation-train gap at best epoch: +0.0389
val/train ratio = 2.62
=> Best generalization so far and a slightly better score

Swin Transformer (Swin-T)

Transformer

768

backbone output size

60 M

total trainable parameters

TECHNICAL PARTICULARITY

- The only model with no classic convolution: the spectrum is split into windows, attention is computed inside each, and the windows shift between layers.
- It works at the spectrum's native resolution: it pads internally to multiples of 7, so no resizing is forced.

KEY CONCEPT

Attention compares distant regions directly, instead of being limited to a local neighborhood the way convolution is.

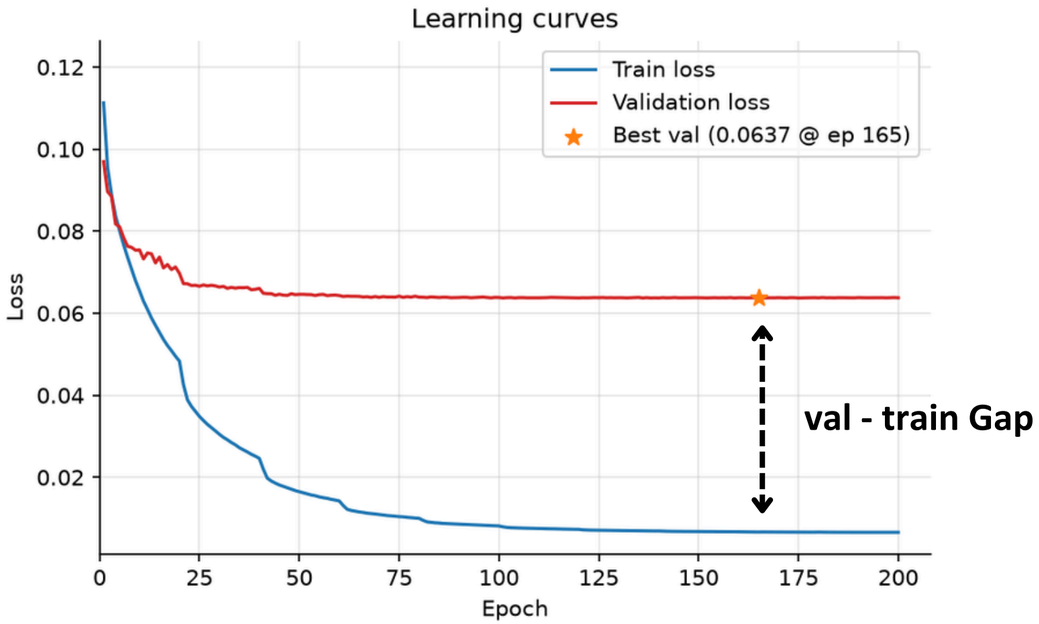
Test objective: Is attention, with no locality bias, well suited to this kind of spectrum?

$$Attention\,(Q, K, V) = SoftMax\left(\frac{QK^T}{\sqrt{d}} + B\right)V$$

Q, K, V: query, key, value => linear projections of the patches inside a window
d: dimension of the query/key vectors, used to scale the dot product
B: relative position bias, specific to each pair of positions within the window

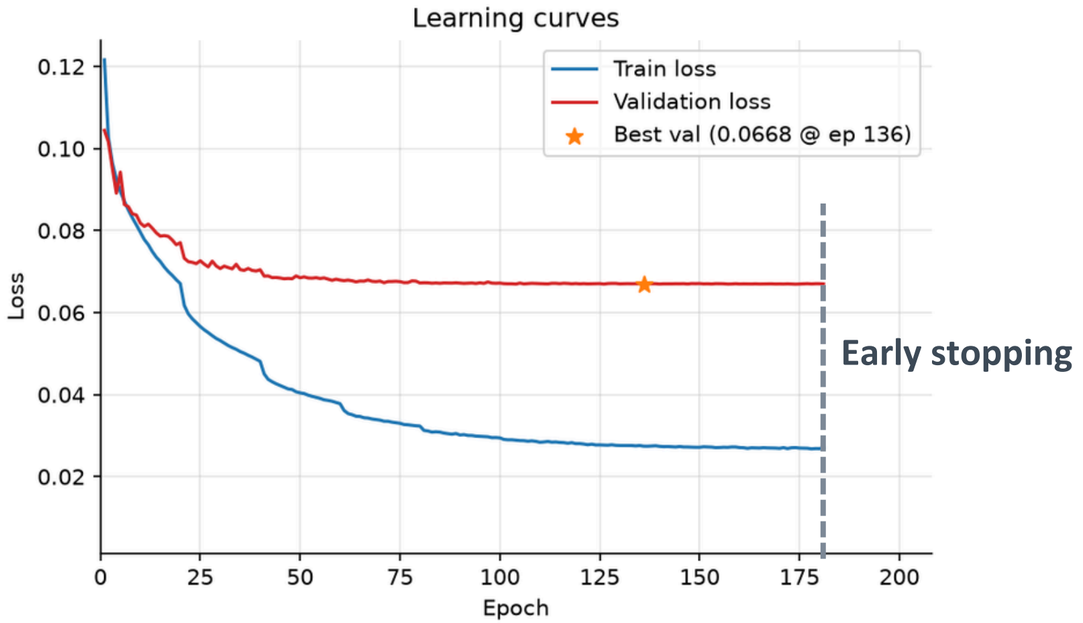
Swin Transformer (Swin-T) vs. Baseline CNN

Baseline CNN



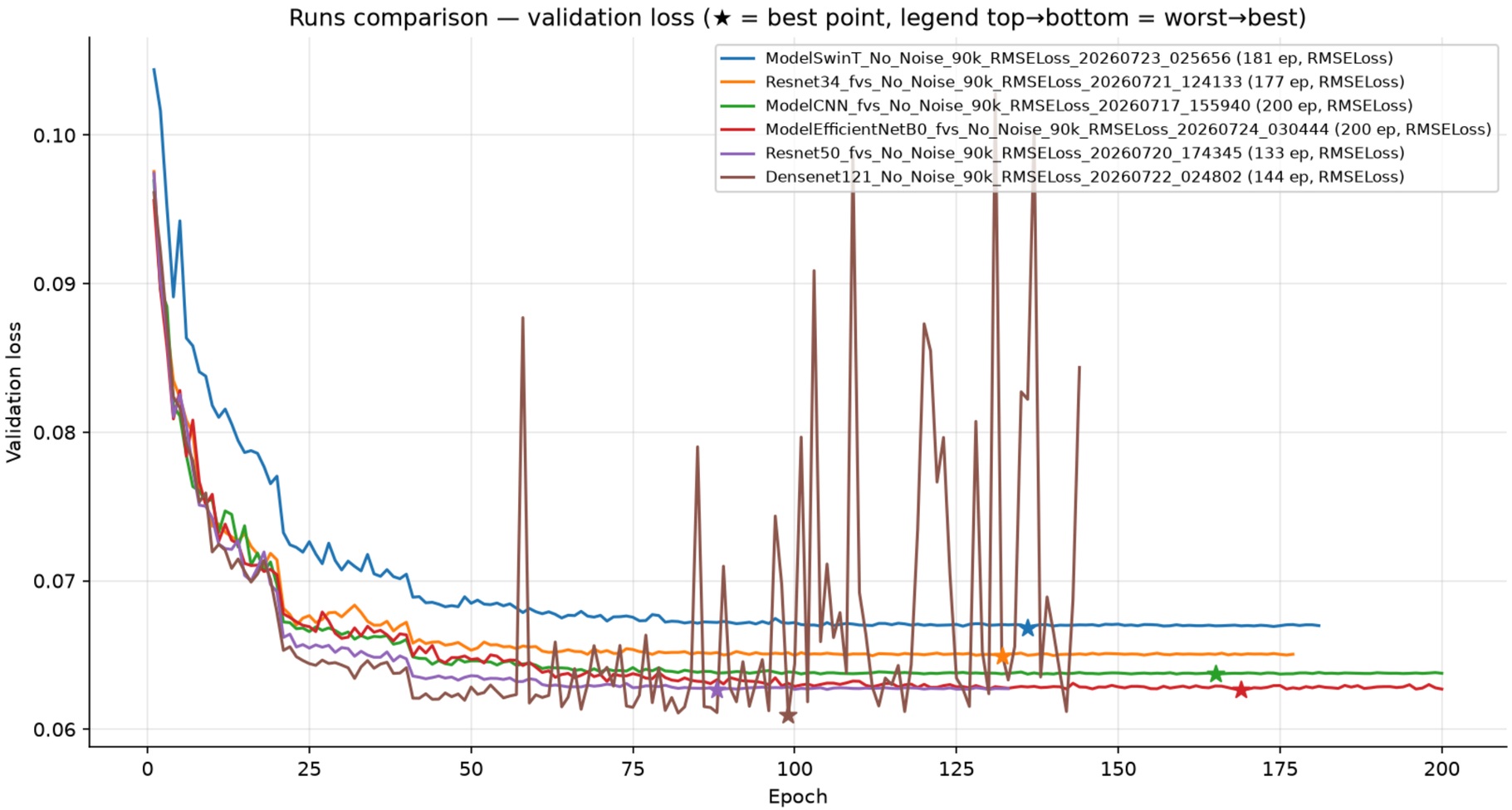
Validation-train gap at best epoch: +0.0570
val/train ratio = 9.57

Swin Transformer (Swin-T)



Validation-train gap at best epoch: +0.0394
val/train ratio = 2.44
=> Comparable validation loss with far less overfitting.

All model comparison - 90K Dataset

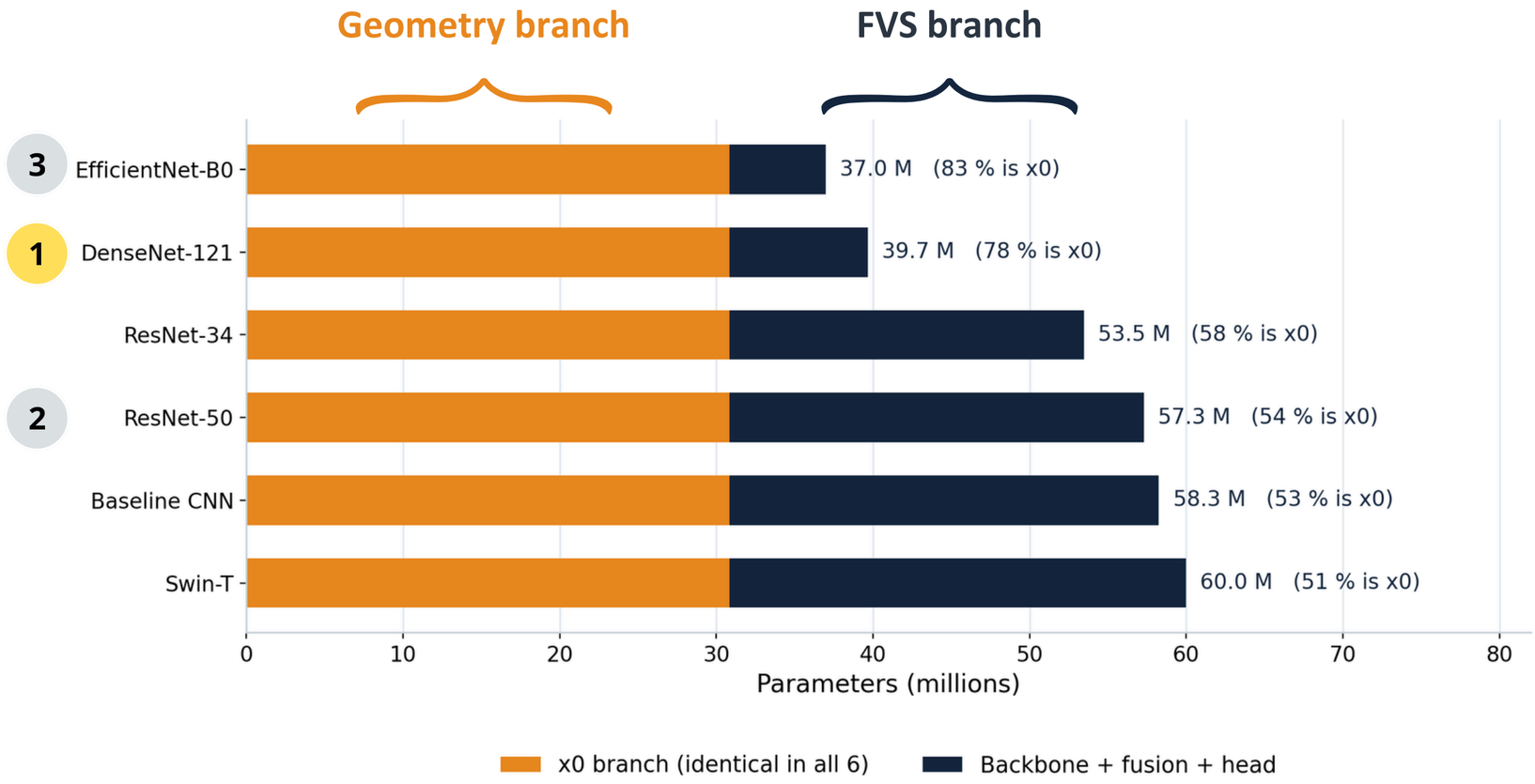


All models overfit and memorize the dataset, adding Gaussian noise does not improve the models.
We need to adjust the complexity of the synthetic data to see if there is any improvement.

Parameter count

30.8 M parameters just for x0!

The geometry branch copies x0 across 14,516 cells, then feeds them to Linear(14516, 2048). These three linear layers hold 30.8 M weights, identical in all six models.



Both the DenseNet-121 and EfficientNet-B0 models have three times fewer parameters for the FVS image while outperforming the baseline.

Comparing 3 models

Model	Family	Backbone output	Total params	Idea tested
Baseline CNN	Residual CNN	1000	58.3 M	x0 encoded by a 30.8 M-param dense branch, **BASELINE**
CNN V2	Residual CNN	1000	27.4 M	x0 as a true scalar : 30.8 M to **4.3 K params**
CNN All geo + FiLM	Residual CNN	1000	27.7 M	full array geometry + FiLM to improve accuracy 30.8 M to **4.6 K params**

All three models share the same spectrum-image branch (ResNet-50 backbone) and the same fusion + prediction head architecture.
The only change is the geometry branch.

ModelCNN_fvs_v2

A tiny scalar MLP replaces the baseline: x0 enters as one value instead of being broadcast to 14,516 copies.

Amplitude matrix size : 76 × 191 = **14,516**

Spectrum image (fvs)
14 516 inputs

Pretrained CNN backbone **(ResNet-50)**
→ **1000 Output**

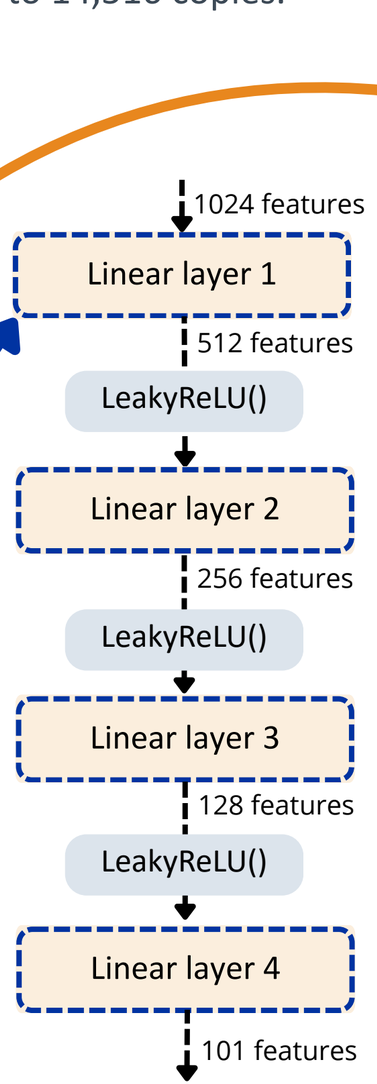
Array geometry (x0')
1 input

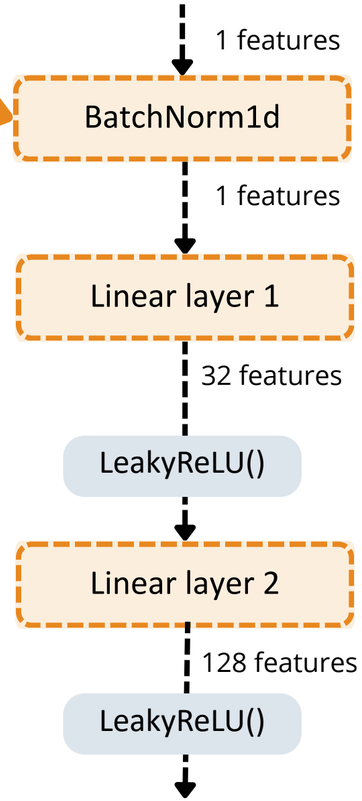
Scalar encoder MLP
→ **128 Output**

Fusion: the two feature vectors are concatenated
1000 + 128 input → 1024 Output

Prediction head dense layers + LeakyReLU()

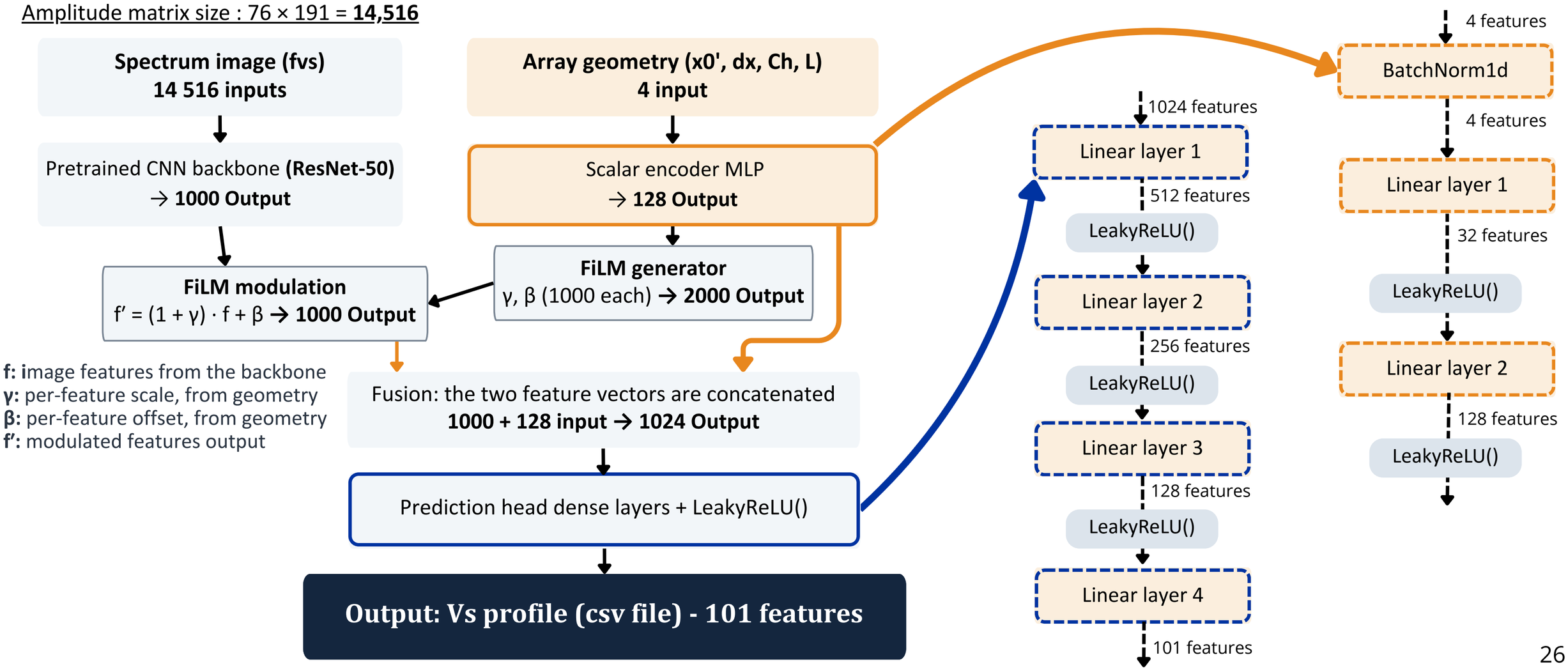
Output: Vs profile (csv file) - 101 features



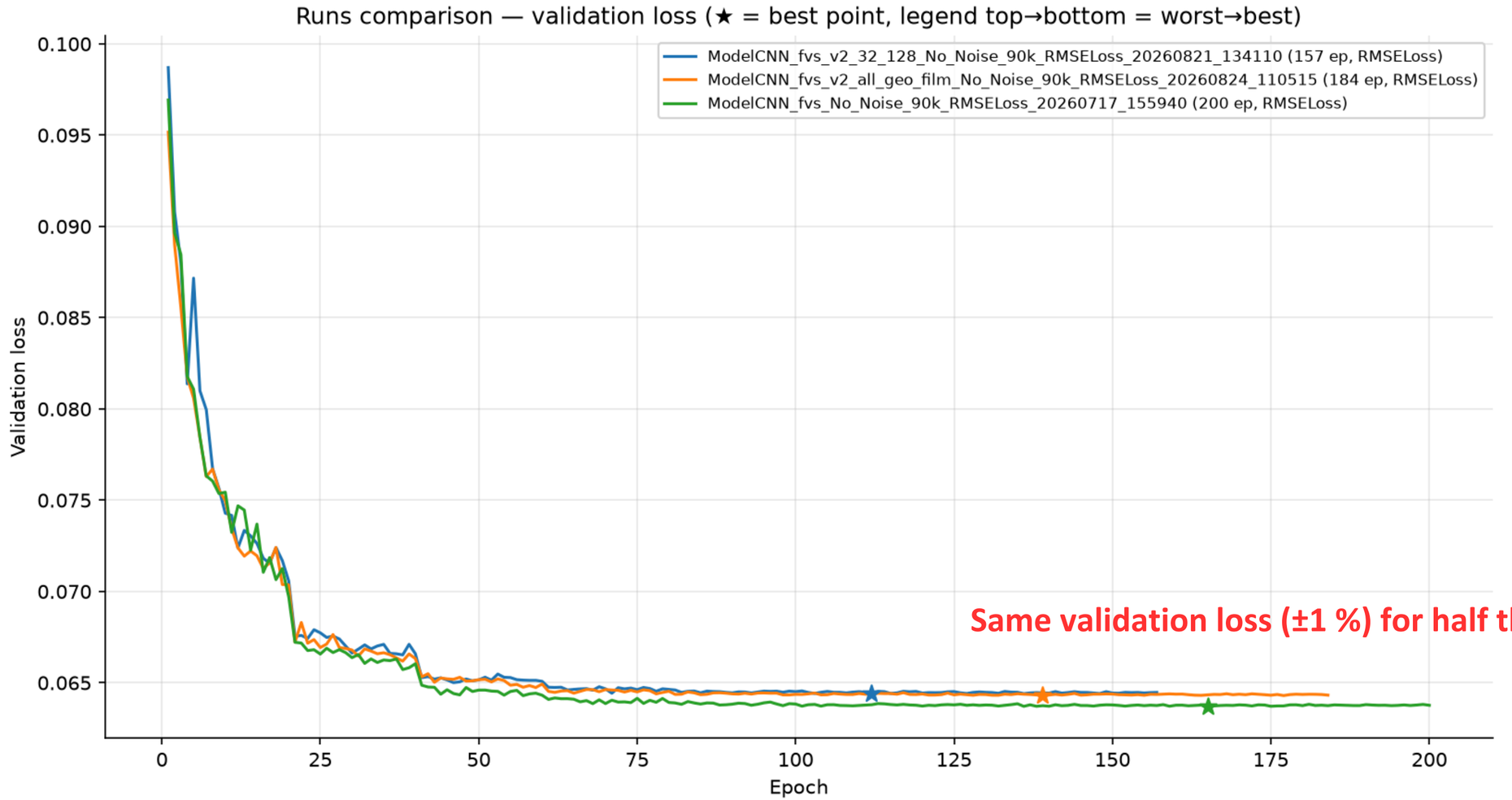


ModelCNN_FiLM + Geometry array

Two branches - the geometry modulates the spectrum features (Feature-wise Linear Modulation), then both are concatenated.



All model comparison - 90K Dataset

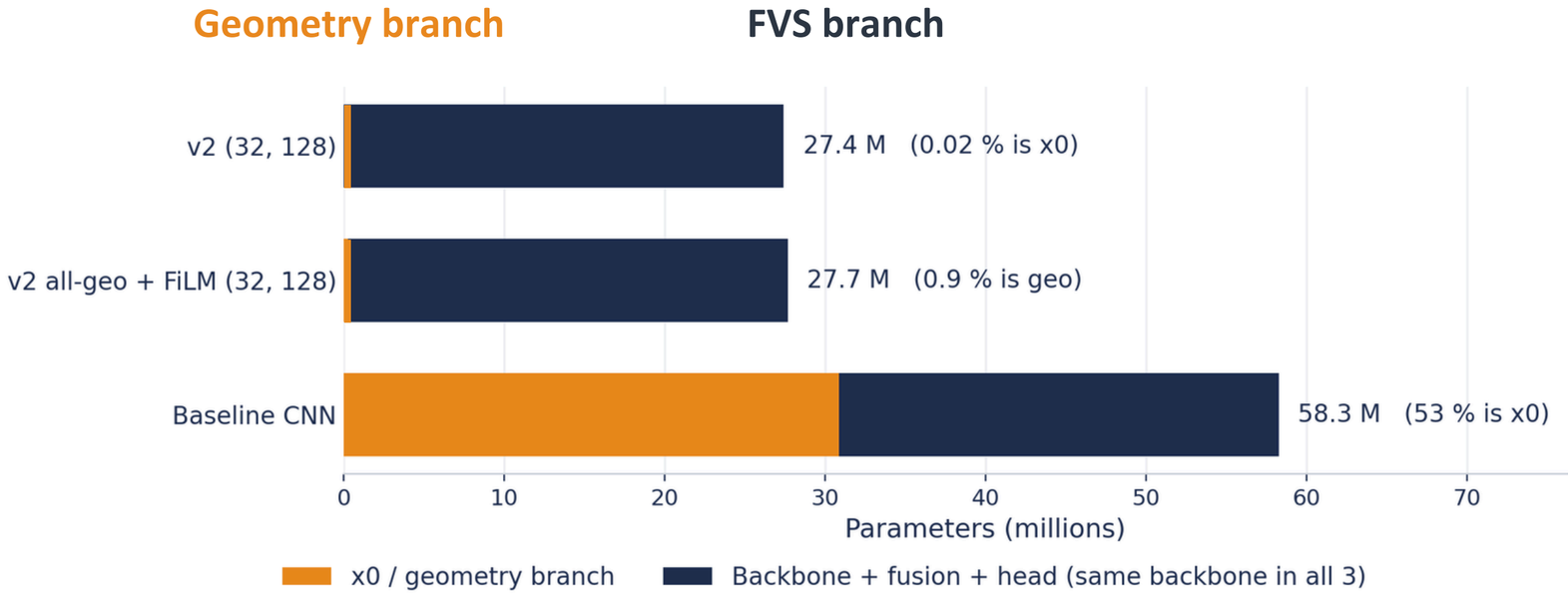


Same validation loss (±1 %) for half the parameters

Parameter count

Down to 0.9 % for the geometry

The scalar encoder maps x0’, dx, Ch and L to 128 features, and FiLM adds 258 K to modulate the image features. The whole geometry branch now costs less than 1 % of the model, the backbone becomes the main model.



The geometry branch drops from 53 % of the model to under 1 %, for the same validation loss (±1 %) and faster epochs.

CONCLUSION

What we can say today

- All six models overfit. Their training loss keeps going down, but their validation loss stops improving early and stays much higher.
- DenseNet-121 reaches the lowest validation loss (0.061) with 3× fewer parameters than the baseline, despite late-training instability spikes.
- EfficientNet-B0 matches ResNet-50's accuracy (0.0627) with 35% fewer parameters and far less overfitting.
- The Swin Transformer generalizes best of all (ratio 2.4) but its validation loss remains the highest.
- Gaussian noise (σ = 0.02-0.1) never improves validation loss, for any architecture tested.
- The baseline spent 53 % of the whole model encoding a single scalar. The MLP branch now takes under 1 %, and reaches essentially the same validation loss.

FUTURE WORK

Interesting avenue to explore

- Mask random frequency bands (5–10 Hz and 60–80 Hz) during training so the model learns from partial spectra (ongoing).
- Diagnose DenseNet-121's instability: test batch normalization settings and batch size to remove the loss spikes.
- Combine the best backbones (DenseNet-121, EfficientNet-B0) with the MLP geometry branch.

- **Bring synthetic data closer to real conditions** with noise-injection augmentation: train on synthetic FVS images blended with real field noise (Zhu, Mousavi & Beroza, 2019 - "DeepDenoiser").

Sources and references

- Tran, Q.K. et al. (2025). "Backcalculation of shear-wave velocity and thickness of asphalt pavement by deep-learning inversion of surface-wave dispersion spectra." International Journal of Pavement Engineering.
- Cho, S., Pyun, S., Choi, B., Lee, G., Jang, S., & Choi, Y. (2024). "Prediction of S-wave velocity models from surface waves using deep learning." Near Surface Geophysics,
- Wu, T.-J. et al. (2025). "Accurate non-destructive evaluation of pavement stiffness profile based on Monte Carlo inversion of surface-wave dispersion spectra." Soil Dynamics and Earthquake Engineering.
- Liu, Z. et al. (2021). "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows." ICCV.
- Tan, M. & Le, Q. (2019). "EfficientNet: Rethinking Model Scaling for CNNs." ICML.
- Huang, G. et al. (2017). "Densely Connected Convolutional Networks" (DenseNet). CVPR.
- He, K. et al. (2016). "Deep Residual Learning for Image Recognition" (ResNet). CVPR.
- Park, C.B., Miller, R.D., Xia, J. (1999). "Multichannel Analysis of Surface Waves (MASW)." Geophysics.
- Zhu, W., Mousavi, S.M., Beroza, G.C. (2019). "DeepDenoiser: seismic signal denoising and decomposition using deep neural networks."

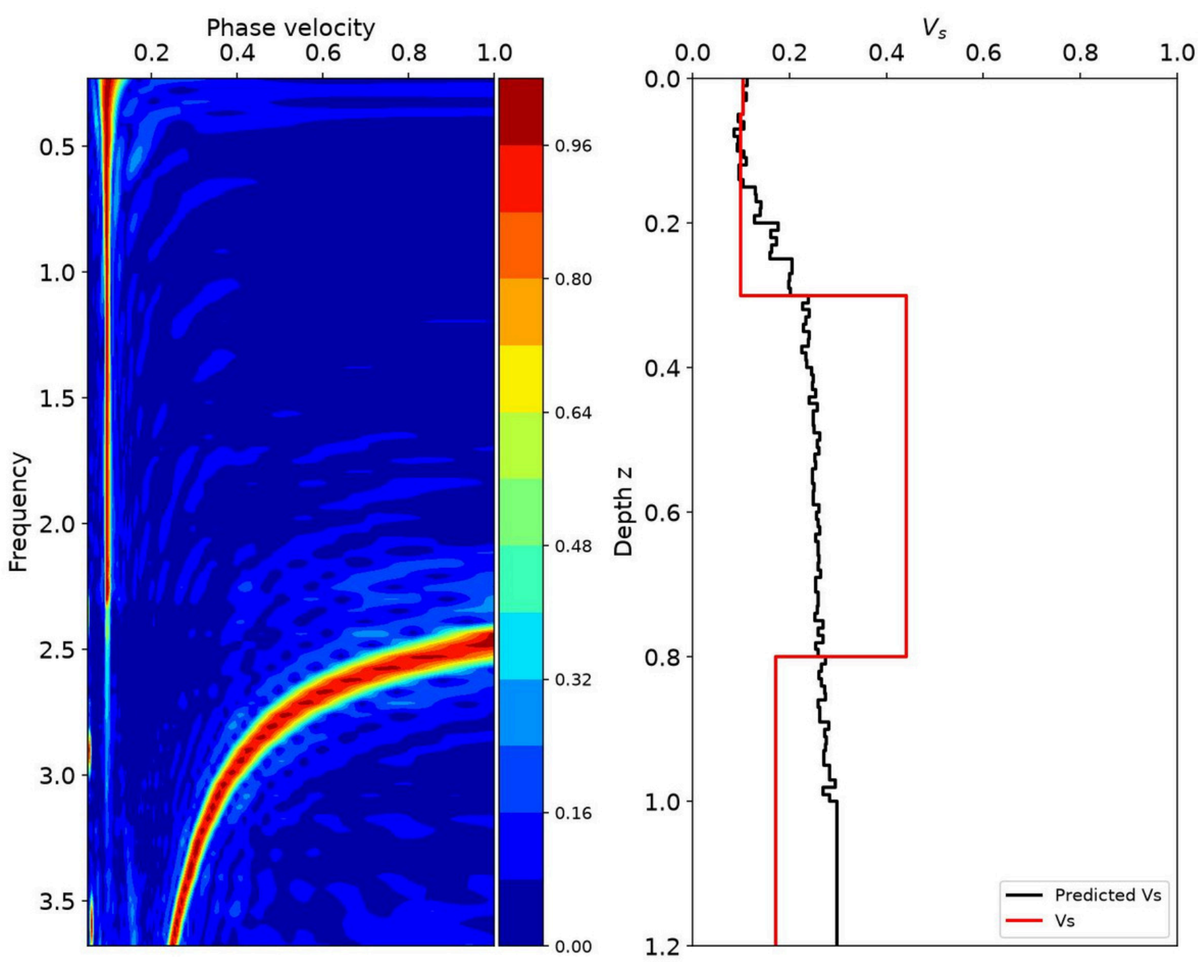
Thank You for Listening

Any questions, comments, suggestions?

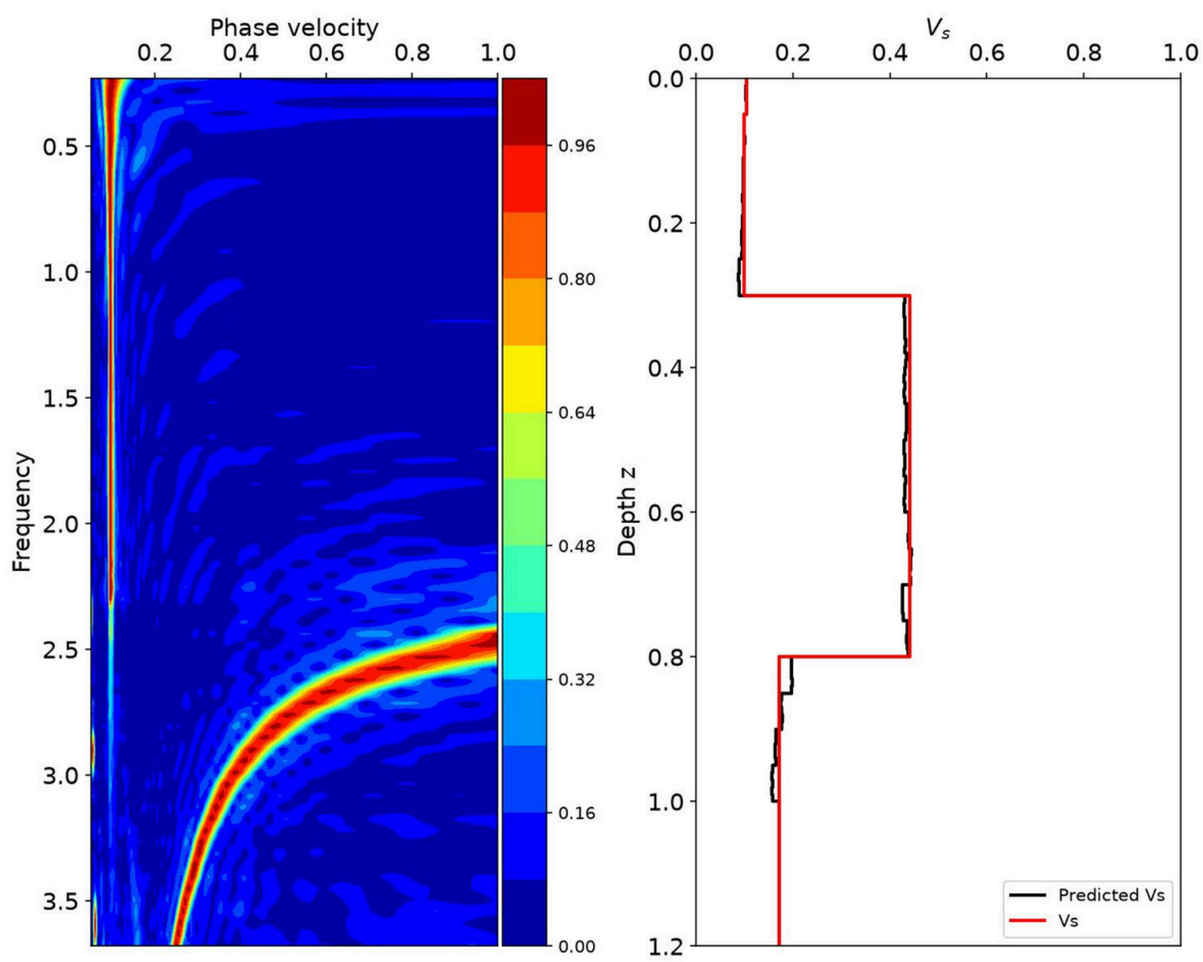
NYCU - Department of Civil Engineering

DenseNet-121: Model100

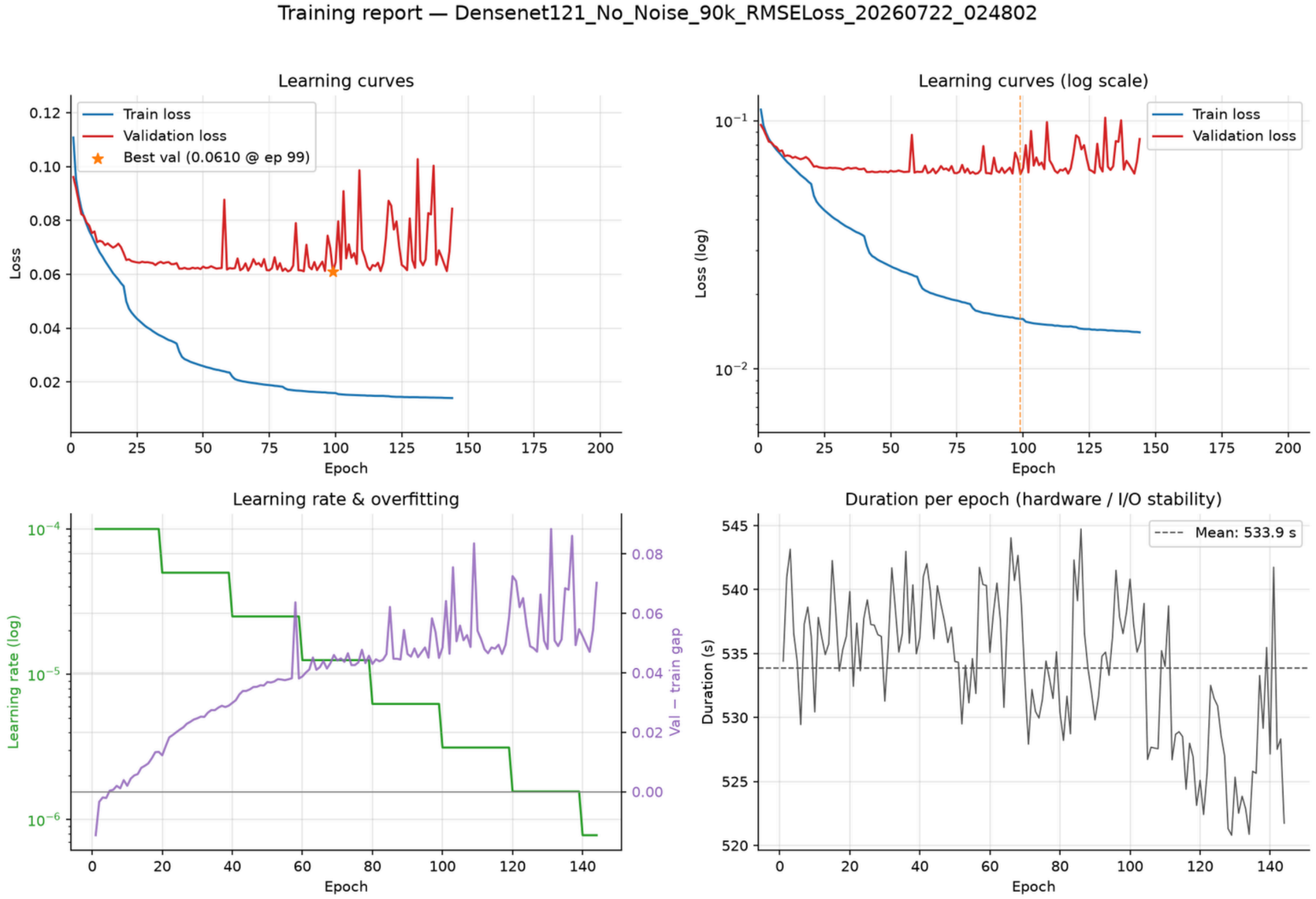
First epoch



Best epoch

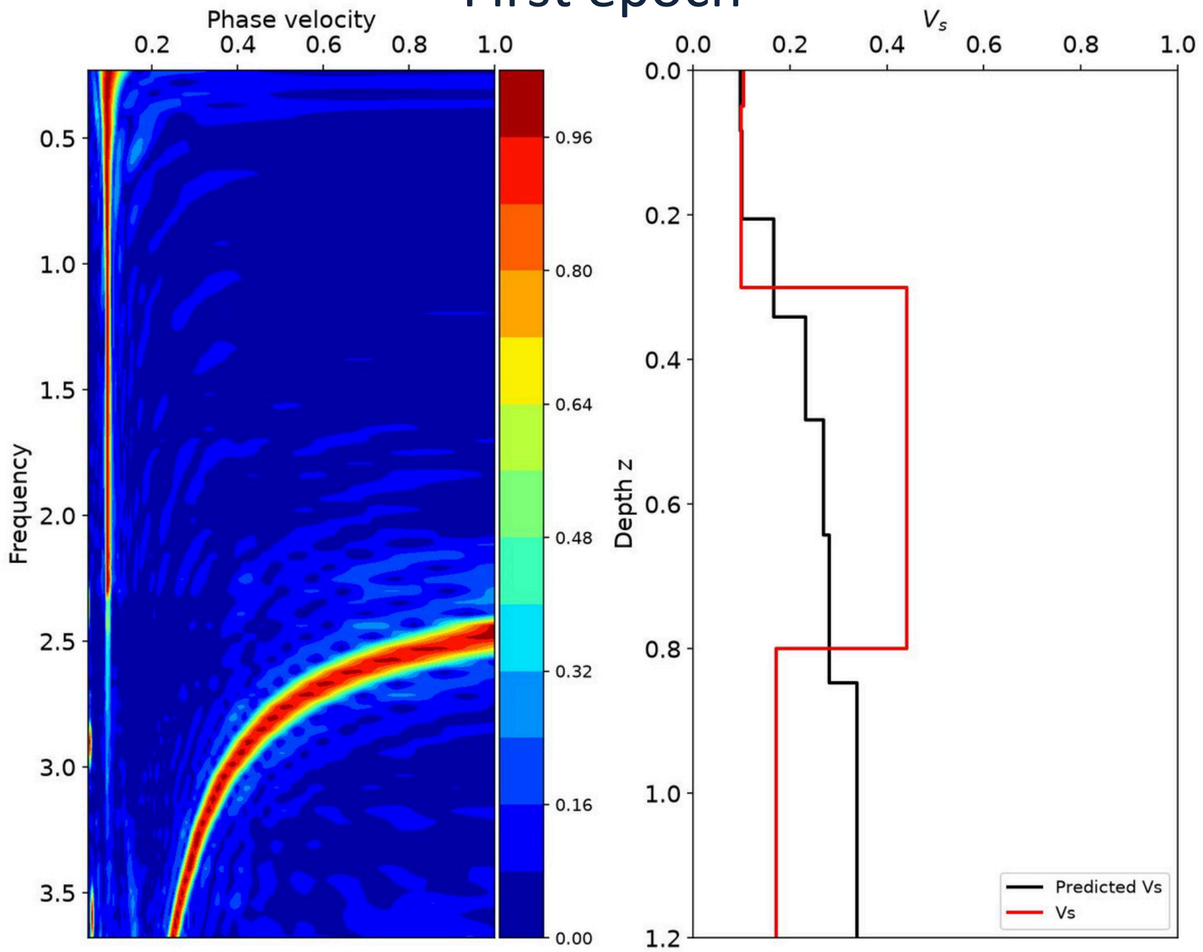


DenseNet-121: Model100

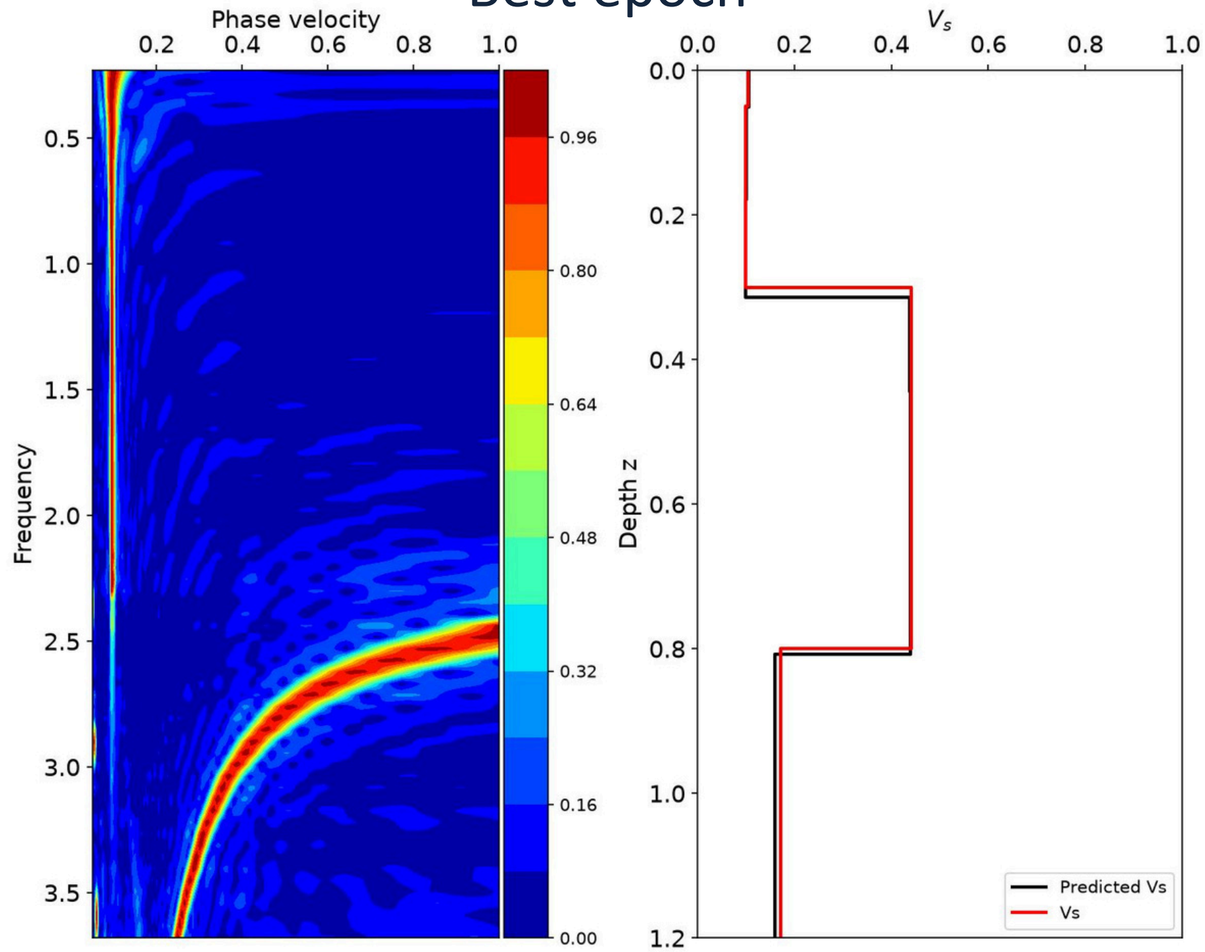


DenseNet-121: Model6

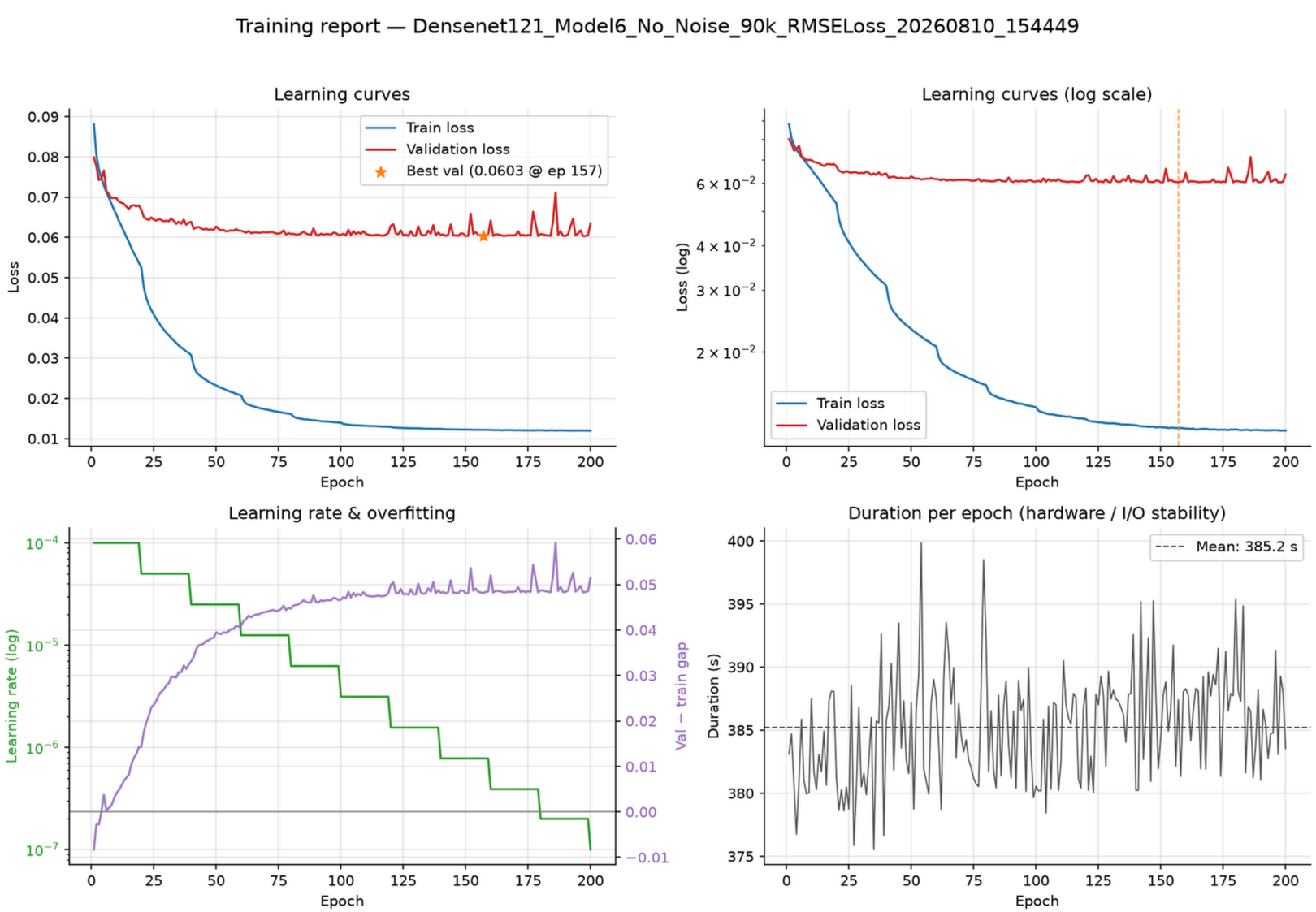
First epoch



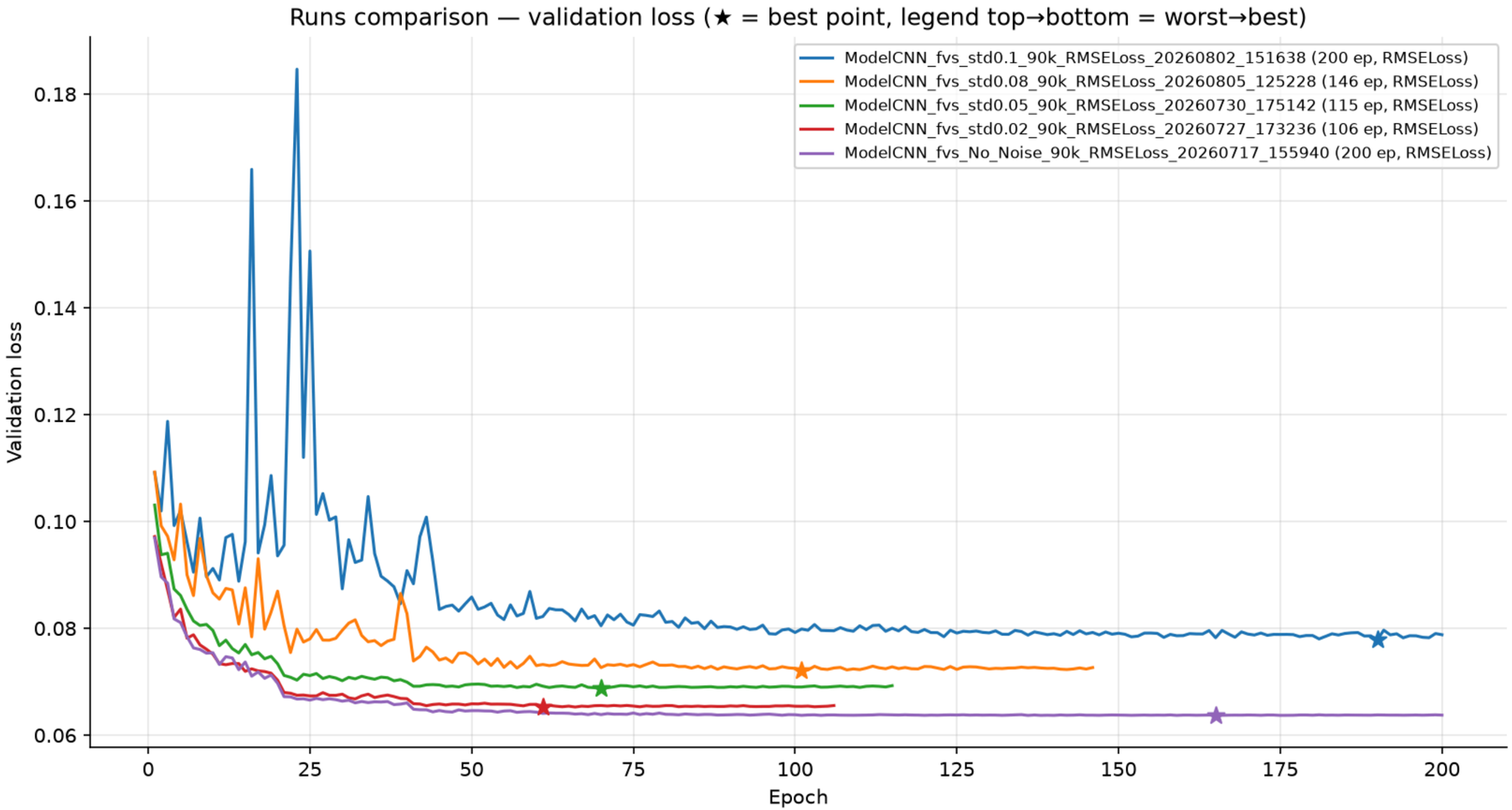
Best epoch



DenseNet-121: Model6

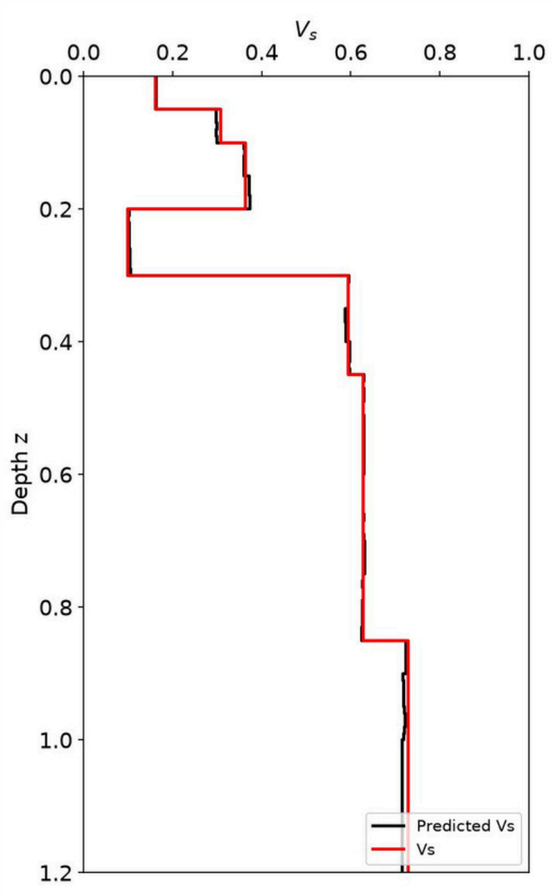


ModelCNN_fvs (Baseline) with Gaussian noise

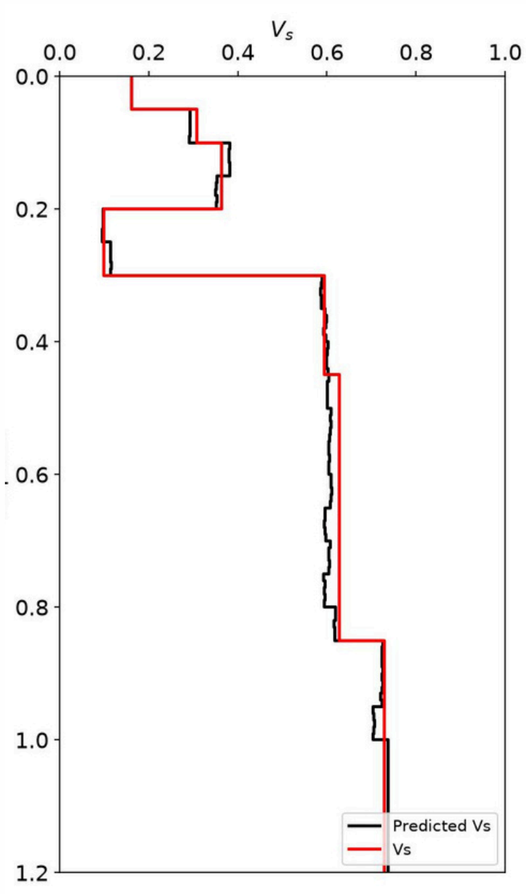


ModelCNN_fvs (Baseline) with Gaussian noise

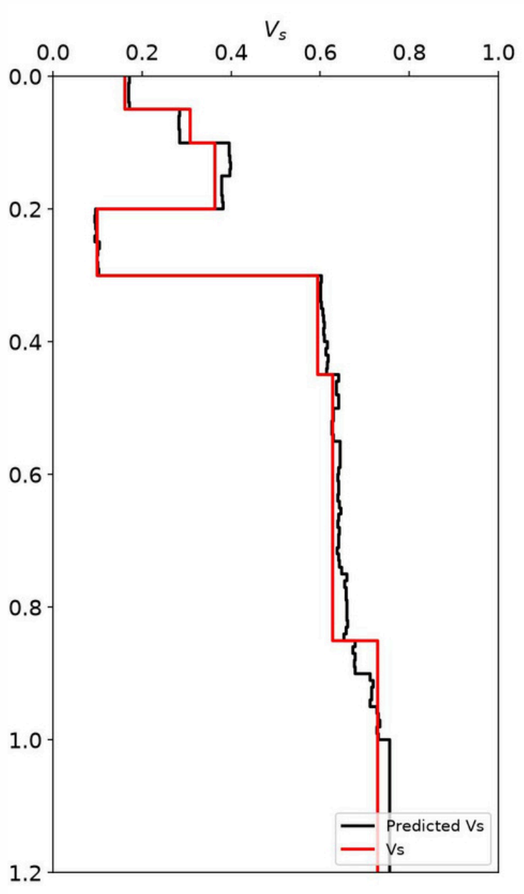
Without noise



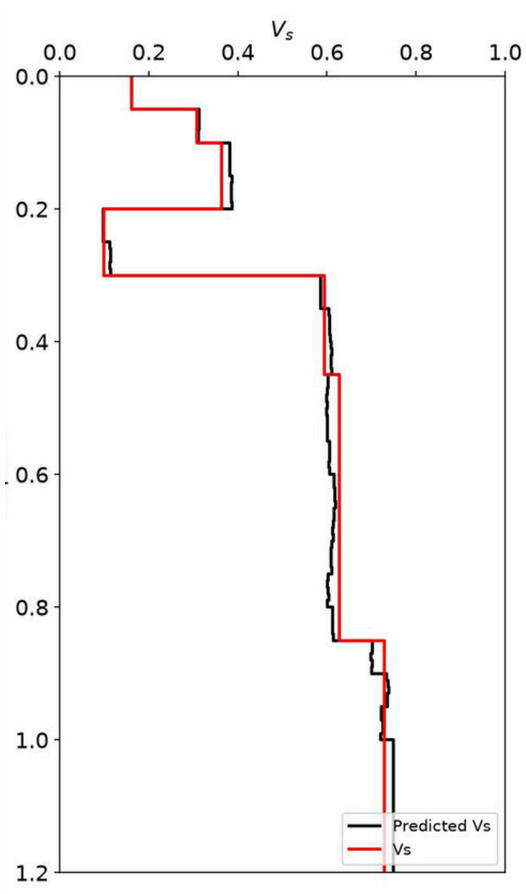
std = 2%



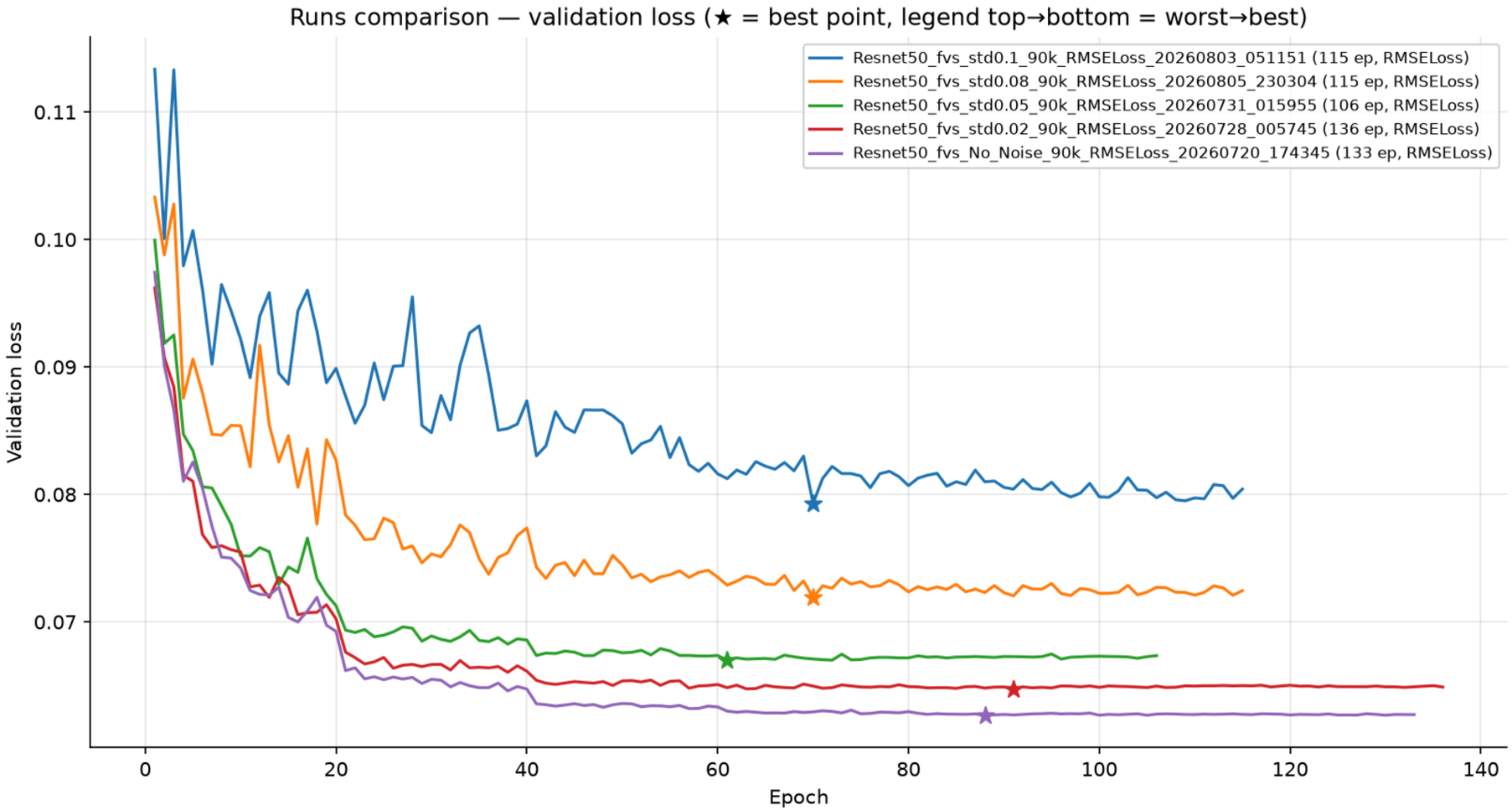
std = 5%



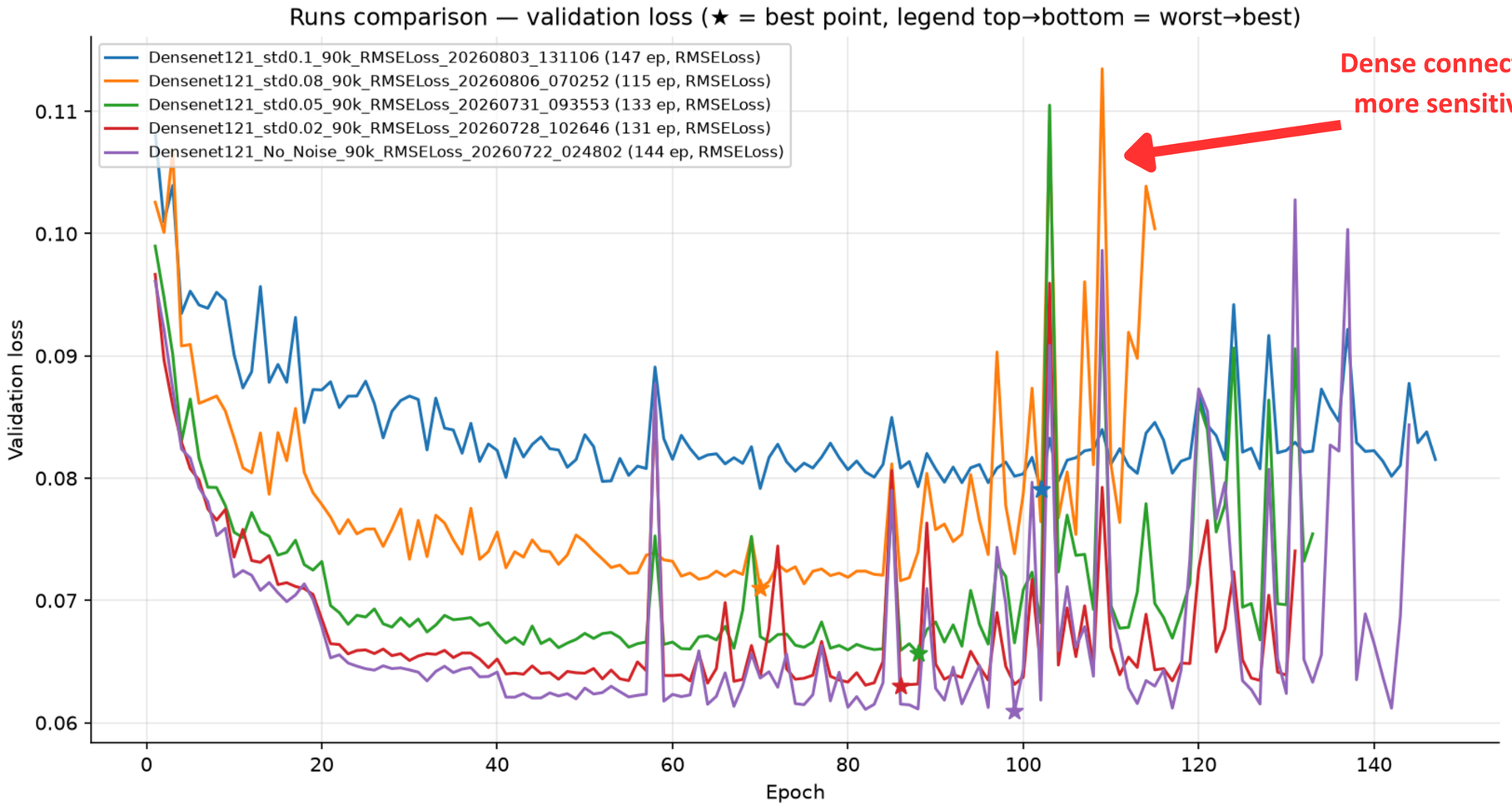
std = 10%



ResNet-50 (revised) with Gaussian noise

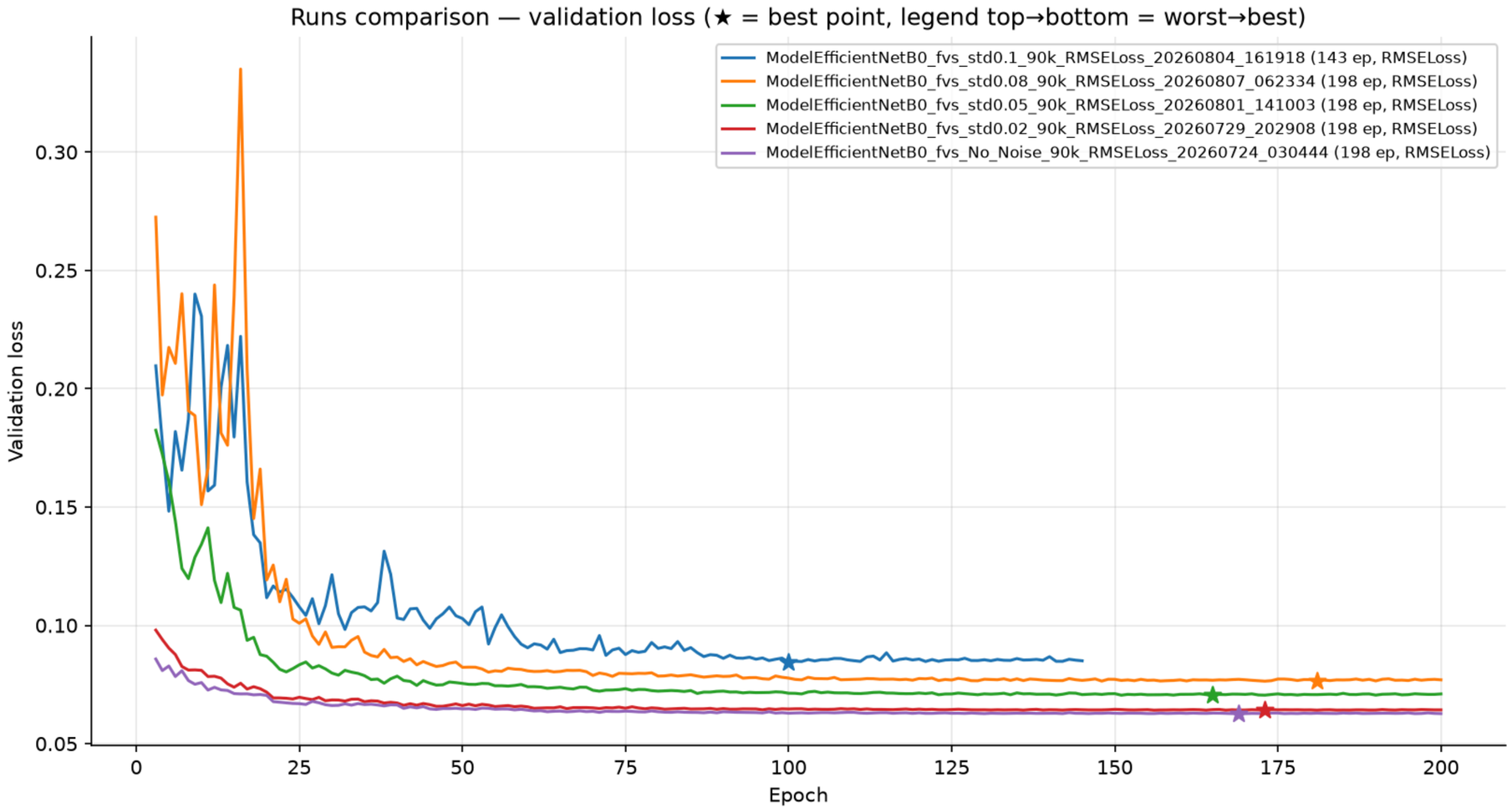


DenseNet-121 with Gaussian noise



Dense connections make BatchNorm more sensitive to the learning rate

EfficientNet-B0 with Gaussian noise



Swin Transformer (Swin-T) with Gaussian noise

