HyTAS: A Hyperspectral Image Transformer Architecture Search Benchmark and Analysis

Fangqin Zhou[1], Mert Kilickaya[1], Joaquin Vanschoren[1], and Ran Piao[1]

Automated Machine Learning Group, Eindhoven University of Technology, the Netherlands
{f.zhou, j.vanschoren}@tue.nl
https://github.com/openml-labs

7 Supplementary

7.1 Further Details of Proxies

Table 1 shows the symbols and equations of proxies.

N	the number of layers
$\mathcal{L}$	the loss function of a neural network with parameters θ_i
$\odot$	the Hadamard product
H	the Hessian-gradient matrix
$\mathcal{S}_z$	the saliency per activation z
M	the length of the vectorized feature map
B	the number of input samples
λ_i	the eigenvalues of the covariance matrix
ϵ	a small value introduced for stability
f	the number of layers of MSA (multi-self attention module)
k	the number of layers of MLP (multi-layer perceptron module)
nuc	the Nuclear-norm
$\mathcal{Z}$	feature consistency between output of clean input (x) and perturbed input (x')
$\mathcal{P}$	consistency of parameters obtained from x and from x'
$\mathcal{G}$	consistency of gradients obtained from x and from x'

Proxy	Scoring function ($\mathcal{F}_s$)
GradNorm	$\sum_{i=1}^{N} \left\| \frac{\partial \mathcal{L}}{\partial \theta_i} \right\|_2$
SNIP	$\sum_{i=1}^{N} \left\| \frac{\partial \mathcal{L}}{\partial \theta_i} \odot \theta_i \right\|$
GraSP	$-\sum_{i=1}^{N} \left(\left(H \frac{\partial \mathcal{L}}{\partial \theta_i} \right) \odot \theta_i \right)$
Synflow	$\sum_{i=1}^{N} \left(\frac{\partial \mathcal{L}}{\partial \theta_i} \odot \theta_i \right)$
LogSynflow	$\sum_{i=1}^{N} \left(\log \left(\frac{\partial \mathcal{L}}{\partial \theta_i} + 1 \right) \odot \theta_i \right)$
Fisher	$\sum_{i=1}^{N} \sum_{j=1}^{M} \left(\frac{\partial \mathcal{L}}{\partial z_j} z_j \right)^2$
JacobCov	$-\sum_{i=1}^{B} \left[(\lambda_i + \epsilon) + (\lambda_i + \epsilon)^{-1} \right]$
DSS	$\sum_f D^f_{MSA} + \sum_k S^k_{MLP}$, i.e. $D^f_{MSA} = \sum_i \left\| \frac{\partial \mathcal{L}}{\partial \theta_i} \right\|_{nuc} \odot \|\theta_i\|_{nuc}$, and $S^k_{MLP} = \sum_i \frac{\partial \mathcal{L}}{\partial \theta_i} \odot \theta_i$
ZiCo	$\sum_{i=1}^{N} \log \left(\sum_{\omega \in \theta_l} \frac{\mathbb{E}[\lvert \nabla_\omega L(X_i, y_i; \Theta) \rvert]}{\sqrt{\mathrm{Var}(\lvert \nabla_\omega L(X_i, y_i; \Theta) \rvert)}} \right)$
CRoZe	$\sum_{n=1}^{N} \mathcal{Z}_n \times \mathcal{P}_n \times \mathcal{G}_n$, i.e. $\mathcal{Z}_n = 1 + \frac{z_n \cdot z'_n}{\|z_n\| \|z'_n\|}$, $\mathcal{P}_n = 1 + \frac{\theta_{1,n} \cdot \theta'_{1,n}}{\|\theta_{1,n}\| \|\theta'_{1,n}\|}$, $\mathcal{G}_n = \left\lvert \frac{g_n \cdot g'_n}{\|g_n\| \|g'_n\|} \right\rvert$
T-CET	$\sum_l^N \frac{S_n^l}{\sigma^l} \cdot \log \left\lvert K^l \right\rvert$, i.e. $S_n^l = \sum_i S(\theta_i)$ and $\sigma^l = \sqrt{\frac{1}{\lvert \Theta^l \rvert} \sum_i \left(S(\theta_i) - \frac{S_n^l}{\lvert \Theta^l \rvert} \right)^2}$

Table 1: Formulas of zero-cost proxies.

7.2 Experimental Details

Implementation. Our framework was implemented with PyTorch. The backbone code was adapted from the framework of training-free transformer architecture search by Zhou *et al*. [7] and the code of Hyperspectral image transformer by Hong *et al*. [2]. Architecture searching and retraining experiments were run on NVIDIA RTX A6000 GPUs. Our code and results are available at HyTAS.

Datasets. The datasets used in our benchmark consist of the following five standard, public Hyperspectral image datasets.

i. Indian Pines [1]: gathered by AVIRIS sensor over the Indian Pines test site in North-western Indiana.
ii. Salinas scene [1]: gathered by AVIRIS sensor over Salinas Valley, California.
iii. Houston2013 [4]: gathered by ITRES CASI-1500 sensor over the University of Houston campus.
iv. Pavia University (PaviaU) [1]: gathered by the ROSIS sensor over Pavia, northern Italy.
v. Kennedy Space Center (KSC) [1]: gathered by AVIRIS sensor over the Kennedy Space Center, Florida.

We summarize the main features of the datasets in Table 2. For fair comparisons with SpectralFormer [2], a manually crafted transformer, we match the numbers of training and testing samples per class used in SpectralFormer for Indian Pines, Houston2013, and PaviaU. Details are provided in Tables 3, 4, and 5, respectively. For Salinas (see Table 6) and KSC (see Table 7) datasets, we randomly sample 50 and 25 samples, respectively, for training, with the remaining samples reserved for testing.

Dataset	**Indian Pines**	**Salinas**	**Houston2013**	**PaviaU**	**KSC**
Spatial Dimension	145x145	512x217	349x1905	610x340	518x620
#Bands	200 (224)	204 (224)	144	103	176
#Wavelength(nm)	400-2500	360-2500	380-1050	430-860	400-2500
#Classes	16	16	15	9	13
#Train Samples	695	800	2832	3921	325
#Test Samples	9671	53329	12197	40002	4886
Domain	Agriculture	Agriculture	Other	Other	Other

Table 2: Dataset details. Indian Pines and Salinas datasets originally contain 224 wavebands. In our experiments, we used 200 and 204 wavebands for Indian Pines and Salinas, respectively, after removing bands covering the water absorption region.

7.3 The effectiveness of taking the sign of model parameters.

As mentioned in Section 5.3 in our main paper, the original Synflow, LogSynflow, and DSS apply the sign of model parameters before calculating proxy scores.

Class No.	Class Name	Training	Testing
1	Corn Notill	50	1384
2	Corn Mintill	50	784
3	Corn	50	184
4	Grass Pasture	50	447
5	Grass Trees	50	697
6	Hay Windrowed	50	439
7	Soybean Notill	50	918
8	Soybean Mintill	50	2418
9	Soybean Clean	50	564
10	Wheat	50	162
11	Woods	50	1244
12	Buildings Grass Trees Drives	50	330
13	Stone Steel Towers	50	45
14	Alfalfa	15	39
15	Grass Pasture Mowed	15	11
16	Oats	15	5
	Total	695	9671

Table 3: Land-cover classes of Indian Pines dataset, together with the training and testing samples for each class.

Class No.	Class Name	Training	Testing
1	Healthy Grass	198	1053
2	Stressed Grass	190	1064
3	Synthetic Grass	192	505
4	Tree	188	1056
5	Soil	186	1056
6	Water	182	143
7	Residential	196	1072
8	Commercial	191	1053
9	Road	193	1059
10	Highway	191	1036
11	Railway	181	1054
12	Parking Lot1	192	1041
13	Parking Lot2	184	285
14	Tennis Court	181	247
15	Running Track	187	473
	Total	2832	12197

Table 4: Land-cover classes of Houston2013 dataset, together with the training and testing samples for each class.

Class No.	Class Name	Training	Testing
1	Asphalt	548	6304
2	Meadows	540	18146
3	Gravel	392	1815
4	Trees	524	2912
5	Metal Sheets	265	1113
6	Bare Soil	532	4572
7	Bitumen	375	981
8	Bricks	514	3364
9	Shadows	231	795
	Total	3921	40002

Table 5: Land-cover classes of Pavia University dataset, together with the training and testing samples for each class.

Class No.	Class Name	Samples
1	Brocoli_green_weeds_1	2009
2	Brocoli_green_weeds_2	3726
3	Fallow	1976
4	Fallow_rough_plow	1394
5	Fallow_smooth	2678
6	Stubble	3959
7	Celery	3579
8	Grapes_untrained	11271
9	Soil_vinyard_develop	6203
10	Corn_senesced_green_weeds	3278
11	Lettuce_romaine_4wk	1068
12	Lettuce_romaine_5wk	1927
13	Lettuce_romaine_6wk	916
14	Lettuce_romaine_7wk	1070
15	Vinyard_untrained	7268
16	Vinyard_vertical_trellis	1807
	Total	54129

Table 6: Land-cover classes of Salinas dataset, together with the number of samples of each class.

Class No.	Class Name	Samples
1	Scrub	761
2	Willow swamp	243
3	CP hammock	256
4	CP/oak	252
5	Slash pine	161
6	Oak/broadleaf	229
7	Hardwood swamp	105
8	Graminoid marsh	431
9	Spartina marsh	520
10	Cattail marsh	404
11	Salt marsh	419
12	Mud flats	503
13	Water	927
	Total	5211

Table 7: Land-cover classes of Kennedy Space Center dataset, together with the number of samples for each class.

The operation of the sign was typically employed to prevent performance decline during network pruning [6]. However, in proxy-based NAS or TAS, where parameters are initialized only once, this transformation may hinder the search process. Comparative analysis between results obtained with (w Sn) and without sign (wo Sn) for these three proxies are presented in Table 8.

Results show improved performance without the sign operation, except for the PaviaU dataset. Further examination reveals that the proxy scores with sign exhibit nearly perfect correlation with embedding dimension while showing minimal correlation with the other three basic factors (see Fig. 6 in our main paper). Additionally, investigation into the impact of sign demonstrates significant score vanishing in deeper layers (see Fig. 1). **This decay suggests that scores from later layers contribute insignificantly to final proxy scores, prompting a critical assessment of the necessity of sign for Synflow, LogSynflow, and DSS proxies in NAS or TAS.**

7.4 Score Disparities between MSA and MLP Modules

As discussed in Section 5.3 in our main paper, given the distinct structures of MSA and MLP modules within a transformer architecture, we hypothesize that the effectiveness of the two modules for proxy scores might differ significantly. To validate our assumption, we separately compute DSS, Synflow, and SNIP scores for MSA and MLP modules. Our analysis reveals a significant score scale difference between MSA and MLP, as shown in the first row of Fig. 2. To address this imbalance, we apply a logarithmic transformation to both module scores. The transformed results demonstrate comparable scores for MSA and MLP modules across all three proxies, as shown in the second row of Fig. 2.

Dataset	*Synflow*				*LogSynflow*				*DSS*			
	OA		ρ		OA		ρ		OA		ρ	
	w Sn	wo Sn	w Sn	wo Sn	w Sn	wo Sn	w Sn	wo Sn	w Sn	wo Sn	w Sn	wo Sn
Indian Pines	0.79	**0.81**	0.60	**0.69**	0.79	**0.80**	0.60	**0.73**	0.80	**0.81**	0.60	**0.69**
Houston2013	0.84	**0.88**	0.53	**0.68**	0.84	**0.87**	0.52	**0.74**	0.84	**0.88**	0.53	**0.67**
PaviaU	0.90	0.90	**0.73**	0.69	0.90	0.90	**0.73**	0.63	0.89	**0.90**	**0.73**	0.70
KSC	0.87	**0.89**	0.71	**0.82**	0.87	**0.88**	0.68	**0.84**	0.89	0.89	0.72	**0.82**
Salinas	0.92	**0.93**	0.91	**0.93**	0.92	0.92	0.89	0.89	0.92	**0.93**	0.91	**0.94**

Table 8: Comparison of Synflow, LogSynflow, and DSS with (w Sn) and without (wo Sn) considering the sign of model parameters. Generally, omitting the sign operation from these proxies yields improved performance. **Thus, it is unnecessary to consider the sign of model parameters when calculating Synflow, LogSynflow, and DSS scores in NAS or TAS.**

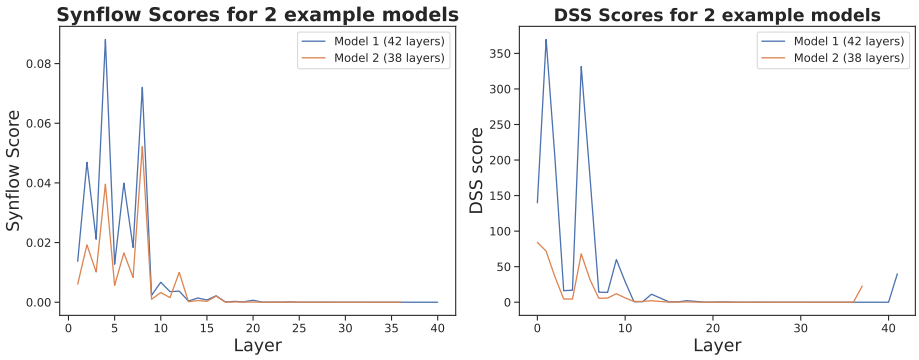

Fig. 1: Synflow (left) and DSS (right) scores after taking the sign of model parameters across layers. Both scores vanish significantly in deeper layers.

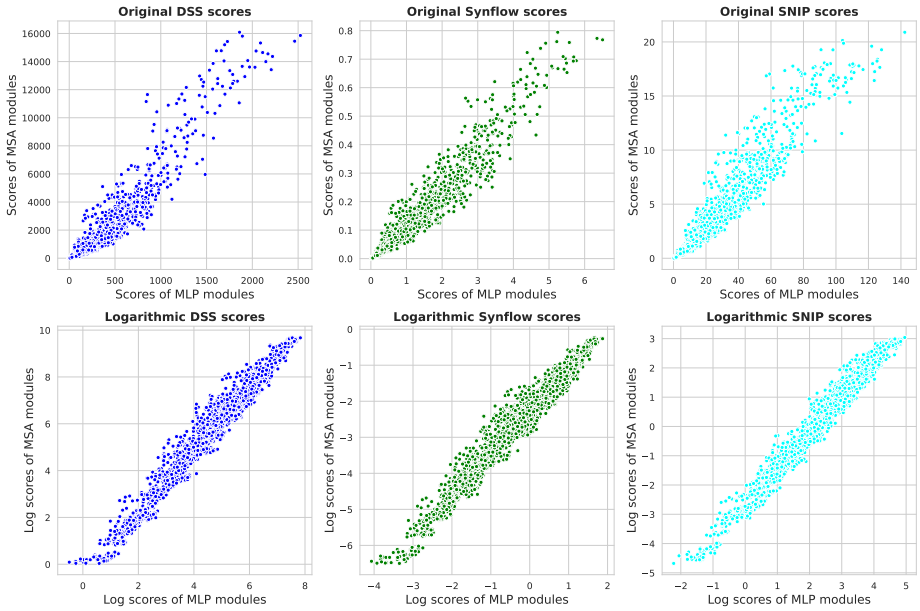

Fig. 2: Discrepancy in proxy scores' scales between MLP and MSA modules for DSS, Synflow, and SNIP proxies before (top row) and after (bottom row) logarithmic transformation. **Original scores show a significant scale difference between MSA and MLP modules, particularly for DSS, where MSA scores are notably higher than MLP. After logarithmic transformation, scores become comparable between the two modules.**

7.5 Experimental Settings of RQ4

The RandomForestRegressor in RQ4 utilizes default settings from the Scikit Learn library [3]. The inputs consist of 2000 samples in total. Each sample represents a subnetwork sampled from the search space, encompassing attributes such as *depth*, *embed_dim*, *num_heads*, *mlp_ratio*, and various proxy scores including SNIP, GradNorm, Synflow, DSS, ZiCo, and Fisher scores. The target variable is OA, derived from the Indian Pines dataset. We randomly split the inputs into training and testing sets with a specified training size.

We repeated each experiment five times with different random seeds to calculate the mean and standard deviation across these five runs. In addition to the result shown in Fig. 8 in our main paper, Fig. 3 illustrates that incorporating proxy scores as input enhances performance, especially with limited training samples. Moreover, fine-tuning the RandomForestRegressor, exploring alternative models, or adjusting proxy selection may further enhance predictive performance.

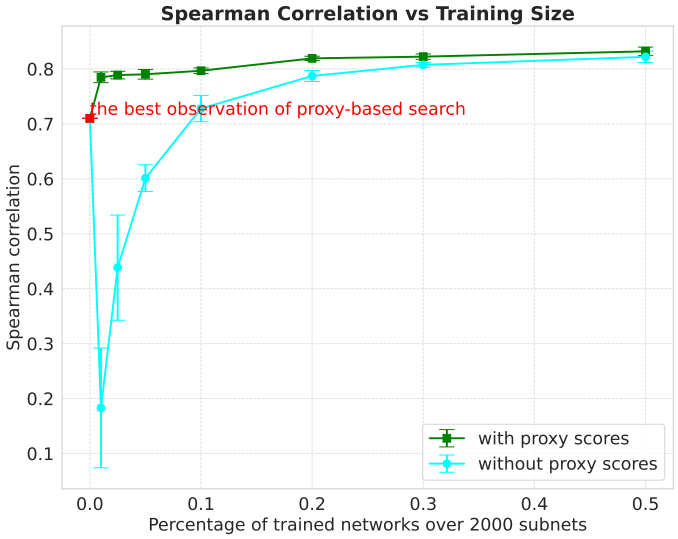

Fig. 3: Comparison of training and testing the Random Forest Regressor with or without proxy scores as input. Proxies include SNIP, GradNorm, Synflow, DSS, ZiCo, and Fisher scores. **Incorporating proxy scores improves performance, particularly with limited training samples (*i.e.*, less than 10% of the total).**

7.6 Differences between RGB and Hyperspectral Domain

We highlight the differences between RGB and Hyperspectral domain to show that designing hyperspectral transformers is non-trivial. Simply adjusting the

input layer of ViT to support higher channels is ineffective. Even with adjusted depth and architectural parameters, ViT achieves only $\sim 70\%$ accuracy on Indian Pines, a 20% reduction from the current SOTA [5]. The reasons are three-fold:

i) Resolution: Hyperspectral images usually have a shallow spatial resolution (e.g., 7×7 vs. 224×224) and high spectral resolution (e.g., 200 vs. 3) compared to RGB images. This disparity requires preserving limited spatial information throughout the network, with most discriminative information in the spectral dimension.

ii) Tokenization: RGB transformers tokenize spatial dimensions by dividing images into patches. Hyperspectral transformers tokenize spectral dimensions, dividing data by spectral bands. This approach focuses on spectral signatures, which are crucial for identifying material properties in hyperspectral images.

iii) Sample Size: Hyperspectral transformers are more compact due to the inherent challenges in collecting and processing hyperspectral data. The acquisition and calibration of hyperspectral images is complex and expensive, leading to smaller available datasets compared to the vast amounts of RGB data. This constraint necessitates designing models that are both efficient and effective with limited data, often resulting in smaller and more specialized architectures that can generalize well despite the limited training samples.

7.7 How to Design a Hyperspectral Transformer?

Based on existing studies and our analysis, we summarize the following guidelines for designing hyperspectral transformer architectures:

1). *Use group-wise spectral embedding and spatial embedding*: To effectively capture the unique spectral-spatial features of hyperspectral images, it is essential to employ group-wise embedding techniques [2]. This involves separating the spectral and spatial dimensions and embedding them individually to ensure that the rich spectral information and the limited spatial information are both adequately represented.

2). *Use attention modules for both spectral and spatial embeddings*: Attention mechanisms are crucial for capturing the intricate spectral-spatial relationships within hyperspectral images. By applying attention modules to both the spectral and spatial embeddings, the model can focus on the most relevant features in each domain, leading to more accurate and robust performance [5].

3). *Incorporate cross-layer feature fusion*: To minimize information loss as the data passes through the network layers, it is important to incorporate cross-layer feature fusion. This technique allows features from different layers to be combined, ensuring that important spectral-spatial information is preserved and utilized throughout the network [2].

4). *Select suitable embedding dimensions and depth*: Embedding dimensions and network depth significantly impact performance. Unlike RGB tasks, deeper architectures are not always better for hyperspectral imaging due to limited data. Large embedding dimensionality is crucial for high performance (see Fig. 4 and Fig. 5 in our main paper). Careful tuning of these parameters is essential while adjusting the number of heads or MLP ratio has minimal impact.

5). *Consider a suitable model size and regularization techniques*: Balance model size to avoid overfitting, especially with limited hyperspectral data, as shown in Fig. 3 in our main paper. Additionally, regularization techniques like dropout, weight decay, or global-local alignment can improve generalization [5].

By following these guidelines, we hope researchers can design hyperspectral transformer architectures that are both efficient and effective, leveraging the unique characteristics of hyperspectral data to achieve superior performance.

7.8 Limitations

We list several limitations of HyTAS as follows:

1. ***Search Space.*** Although our search space encompasses a wide range of architecture selections, it can be further expanded. For instance, the depth and embedding dimension can be extended to larger values with smaller intervals. Additionally, increasing the number of sampled subnetworks beyond 2000 would enable more comprehensive comparisons.
2. ***Datasets.*** While we conducted experiments on five well-established public hyperspectral image datasets, they vary in size and target vegetation domains. However, the benchmark could benefit from including more diverse hyperspectral datasets.

References

1. Graña, M., Veganzons, M., Ayerdi, B.: Hyperspectral remote sensing scenes. Hyperspectral Remote Sensing Scenes - Grupo de Inteligencia Computacional (GIC) `https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes`
2. Hong, D., Han, Z., Yao, J., Gao, L., Zhang, B., Plaza, A., Chanussot, J.: Spectralformer: Rethinking hyperspectral image classification with transformers. IEEE Transactions on Geoscience and Remote Sensing **60**, 1–15 (2021)
3. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
4. Xiong, Z., Minshan, C., Abhinav, S., Díaz, D.J.C.F.: 2013 IEEE GRSS Data Fusion Contest – Fusion of Hyperspectral and LiDAR Data (2013), `https://hyperspectral.ee.uh.edu/?page_id=459`
5. Zhou, F., Kilickaya, M., Vanschoren, J.: Locality-aware hyperspectral classification. arXiv preprint arXiv:2309.01561 (2023)
6. Zhou, H., Lan, J., Liu, R., Yosinski, J.: Deconstructing lottery tickets: Zeros, signs, and the supermask. Advances in neural information processing systems **32** (2019)
7. Zhou, Q., Sheng, K., Zheng, X., Li, K., Sun, X., Tian, Y., Chen, J., Ji, R.: Training-free transformer architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10894–10903 (2022)