Backbone Architectures for Space Domain Awareness

Kyle Merry
Sandia National Laboratories
John Ossorgin, Zach Mekus
Sandia National Laboratories

ABSTRACT

The computer vision community continues to develop new architectures and methods for image processing with neural networks, but not all these advancements are transferable to Space Domain Awareness (SDA) imagery. Many of the popular neural network components and best practices, such as choices in normalization and preprocessing, can have negative effects on a neural network's ability to process SDA data. Of particular interest are decisions in the feature extraction, or "backbone" layers of a neural network, which can prevent it from capturing the salient information in an SDA scene. In this paper, we analyze several popular architecture choices, observe their effects on a model's performance against SDA image processing tasks, and make recommendations for designing backbone architectures for processing SDA data.

1. INTRODUCTION

Neural networks provide state-of-the-art performance for many common image processing tasks, such as detection, segmentation, classification, and de-noising. However, most of the publications and improvements to these models focus on types of imagery common on the internet. Imagery of people, buildings, and vehicles may vary in content, but have similar properties from an image processing perspective. Neural network architectures codify assumptions about the data that they process, such as scale, pixel value distribution, feature size, and object shapes. As a result, many of the publically available backbone models include components that are not suited to Space Domain Awareness (SDA) imagery, and can prevent a network from learning to complete image processing tasks. By identifying these components and potential alternatives, we can modify advanced image processing architectures to improve their suitability to SDA.

In comparison to imagery from social media or similar sources, SDA images have high dynamic range and are very sparse, consisting primarily of background pixels with occasional small, thin objects that occupy few of the pixels in their bounding box. Sparse, high dynamic range imagery interacts poorly with the most popular normalization layers. Small, unresolved objects are not prioritized by neural network models that are designed to detect large, well-resolved objects. Neural networks for SDA must survey large, high-resolution images while perceiving small, pixel-scale features.

In this paper, we consider components that are common in backbone models designed for ImageNet-like and remote sensing data, such as preprocessing, attention, and normalization layers, and aspects such as receptive field. We analyze the suitability of these components to processing SDA data and identify practices that significantly hinder or enhance performance. We provide recommendations for modifying popular image processing backbone models to improve their performance on SDA data, and a recommended backbone model architecture for general-purpose SDA image processing.

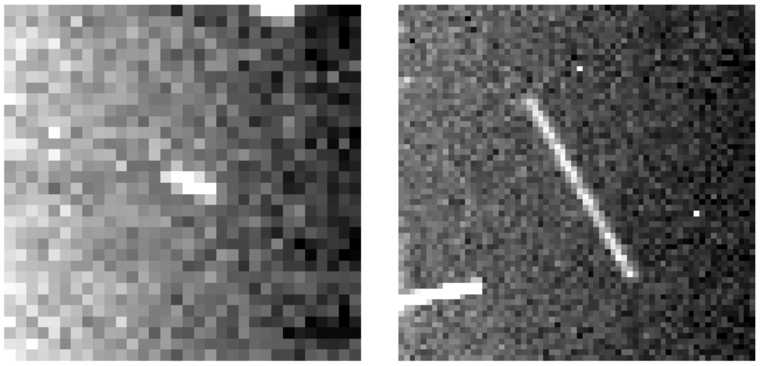

(a) Slow moving RSO (b) Fast moving RSO

Fig. 1: Synthetic imagery exemplar of Resident Space Objects (RSOs) with different apparent velocities

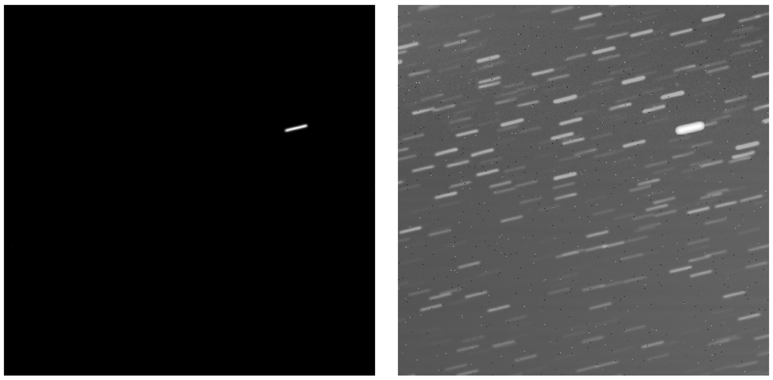
1.1 Unresolved Targets

In ImageNet-like datasets[3], the objects of interest are well resolved in the image, with potentially tens of thousands of pixels sampling a single object. From an image processing perspective, the challenge of detecting these objects is to aggregate information across a large area of the input image, analyzing shapes and textures and their relationships to each other. The object pixels often account for a substantial fraction of the pixels inside of a rectangular bounding box around the target. This allows simple down-sampling operations, such as rectangular pooling kernels, to effectively summarize the information about an object[13]. As a result, convolutional networks designed for object detection on these types of datasets are focused on aggregating information from pixel-scale to large scale (often 1:32 resolution) [7], with pooling and striding operations that hierarchically down-sample the image in rectangular regions. Fine spatial detail is less important than fusion of thousands of pixels.

Wide-area SDA imagery, on the other hand, are characterized by small, point sources that may be moving across the imager's field of view. The Resident Space Objects (RSOs) in the scene are smaller than a single pixel, and are not "resolved" by the sensor into an image of the object, but do reflect light toward the sensor. Blurring of the reflected light due to atmospheric seeing and the optical system results in a characteristic "blob" shape of an unresolved target. For such a target, there is no signal to be found more than a few pixels away from the target's center. For layers of a neural network that was designed to aggregate information across thousands of pixels, the only information to be gained at this scale is context. Context can include the background level, scene conditions, or information about the star field in a catalog-free detection task. We hypothesize that networks should be modified to emphasize high-resolution features, rather than large-stride features.

The behavior is more complicated for fast-moving unresolved RSOs. An unresolved object with apparent motion relative to the imager will appear as a line or "streak" across the image. Depending on sensor configuration and the object's orbit, these streaks can be hundreds or thousands of pixels in length. To be able to simultaneously perceive both ends of a 100-pixel streak, a convolutional neural network would need a receptive field more than 100 pixels across [13]. A convolutional layer with 100 pixels receptive field aggregates information across a $100 * 100 = 10,000$ pixel-square area, with the streak comprising a small fraction. Attention-based models, on the other hand, will be able to aggregate information across the entire image without a rectangular downsampling prior[4][12][1]. We hypothesize that attention-based models will be more effective than convolutional pooling or striding models at streak-based tasks.

These architectures are discussed in detail in Section 2.



(a) No Dynamic Range Compensation (b) IQR-Log Dynamic Range Compensation

Fig. 2: A synthetic SDA scene depicting a noisy star field. Without dynamic range compensation, the brightest star dominates the scaling of the image. While this is an effect of the visualization of the image, it has a similar effect on the normalization layers inside of a neural network. With nonlinear dynamic range compression, we can emphasize the lower end of the representable range and prevent the bright star from suppressing other signals.

1.2 Dynamic Range Compensation

Imagery captured with a consumer camera, such as a smartphone camera, generally has small bit depth (8 bits per channel) and has been pre-processed such that the pixel values are well distributed across the representable range. The brightest possible value is 255 times as bright as the smallest representable value, and about 2 times as bright as a median value.

SDA imagery are generally captured by 12-16 bit imagers in a raw data format, with the brightest representable value potentially 65,000 times brighter than the smallest representable value. This can complicate detection of faint RSOs, since the pixel values for a faint signal are likely near the lower end of the representable range. From a filtering perspective, it is difficult to suppress a signal that is orders of magnitude stronger than the signal of interest with piecewise linear operations such as convolutions and ReLU[5] nonlinearities. Common normalization methods that involve mean and variance computations are not robust to outliers, and are disproportionately affected by the presence or absence of bright stars. As a result, neural network normalization layers such as BatchNorm[9], LayerNorm[2], and GroupNorm[22] can behave differently depending on the brightness of stars in the scene.

We investigated multiple strategies for dynamic range compensation, including modifying the neural network layers and applying dynamic range compression as a pre-processing step. The strategies are described in Section 2.2.

2. METHODOLOGY

2.1 Models Under Test

To establish a baseline of performance on several image processing tasks, we selected a handful of architectures that are popular for SDA, popular in the wider computer vision community, or that we have successfully applied to detection on SDA data.

- **FullRes[14]** A simple convolutional neural network architecture designed for real-time embedded inference on a Field Programmable Gate Array (FPGA). With no upsampling or downsampling layers, the image is processed at full resolution through the entire network. It is simple and effective in low-memory environments, but cannot be scaled to large numbers of parameters.

- **U-Net[15][21]** A family of convolutional neural networks designed for high-resolution tasks, such as segmentation or inpainting. Characterized by a symmetrical, U-shaped architecture with a decoder that mirrors the encoder.

- **Residual Networks (ResNet)[7]** Arguably the most successful convolutional neural network architecture, ResNets are characterized by their use of residual or "skip" connections which stabilize training of very deep models. A ResNet's multi-stage encoder produces a feature pyramid at different resolutions (1:4, 1:8, 1:16, 1:32).
- **Shifted-Window Transformers (Swin)[12]** A ResNet-like network that uses windowed-attention transformer layers instead of convolutions. Architecturally similar to a ResNet, the Swin Transformer also produces a feature pyramid at multiple resolutions.
- **Vision Transfomers (ViT)[4]** Fundamentally different from a convolutional network, ViT is a benchmark for the attention-based models. After breaking the input image in to small tiles, or "patches", the transformer uses global attention to collect information from the entire image simultaneously. This information is used to transform the input tokens into a more useful output representation. Very powerful, scalable models that are limited by computational complexity on large images, since the self-attention computation scales quadratically with the number of tokens.
- **Cross-Covariance Image Transformers (XCiT)[1]** Similar to a ViT, but replaces the token-based self attention with feature-based self attention. As a result, the computational and memory requirements of XCiT scale linearly with the number of input patches, allowing high-resolution processing of very large images.

We used the `timm`[20] implementations for ResNet, Swin, ViT, and XCiT, and our own implementations of FullRes and U-Net.

Model Scaling With the exception of FullRes, we scaled all models to use approximately 10M parameters. Parameter count is an imperfect but commonly used proxy for the learning capacity of a model. 10M parameters is a small configuration for many of these architectures, but our goal is to study the relative performance with respect to different architectural decisions, rather than absolute performance of any particular model. With modern machine learning architectures, higher-parameter-count models nearly always outperform their smaller siblings given enough data and training time[7][12][4]. This is especially true of the feature extraction backbones, where well-trained larger models lead to improved performance on downstream tasks, such as detection or classification[4].

Note that the models have widely varied computational requirements, measured in floating point operations (FLOPs) per image, despite similar parameter count. This can be an important consideration for processing large amounts of data, or for systems with real-time processing requirements.

Table 1: Configuration parameters for model architectures under test.

Parameter	U-Net	FullRes	ResNet+FPN	Swin+FPN	XCiT	ViT
Depths	4 scales	7	(3, 3, 3, 3)	(2, 3, 4, 4)	12	12
Base Dimension	96	64	32	48	256	256
Output embedding dimension	96	64	256	256	256	256
Feature map resolution	1:1	1:1	1:4	1:4	1:8	1:8
Maximum Feature Stride	1:16	1:1	1:32	1:32	1:8	1:8
Parameters	10.7M	60K	10.1M	10.0M	11.6M	10.6M
FLOPs per 1K x 1K image	450B	64B	68B	59B	156B	1,800B

Feature Map Resolution For the ResNet and Swin backbones, we use a Feature Pyramid Network (FPN)[10] neck to produce a high-resolution feature map. Recall that the feature pyramids produced by a model like ResNet or Swin include high-resolution, low-information feature maps from early stages of the network as well as low-resolution, high-information feature maps from the later stages. The FPN module fuses information from the later feature maps back into the earlier ones, producing high-resolution, high-information feature maps. This is conceptually similar to a U-Net, but with a non-symmetrical, encoder-heavy architecture.

2.2 Dynamic Range Compensation

We investigated multiple strategies to compensate for the high dynamic range of SDA data, including modifications to layers within the network and applying dynamic range compression as a pre-processing step.

ReLU vs ReLU6 The Rectified Linear Unit (ReLU)[5] is a popular activation layer in convolutional neural networks. The ReLU6 variant [8] adds an additional nonlinearity, saturating values above some fixed threshold. It is possible that this saturating behavior will block large signals early in the network and prevent them from propagating to the deeper layers, improving performance in high dynamic range situations.

Normalization Layers Batch Norm[9] is a standard normalization for convolutional neural networks, but its sensitivity to training batch size has spurred development of newer normalization methods, including Group Norm[22], Instance Norm[19], or Layer Norm[2]. While the Batch Norm parameters are fixed after training, the newer normalization layers recompute mean or variance estimates at inference time. This can lead to wildly varying behavior based on the presence or absence of bright stars in an image, as demonstrated in Section 3.4

Preprocessing Normalizing the input imagery is common practice with computer vision models, often with mean/variance standardization. We can also manipulate the dynamic range of the data by preprocessing the images before passing them to the model. We experimented with multiple normalization and dynamic range compression strategies, including:

- **Mean/Variance Standardization:** Compute the mean μ and standard deviation σ of every pixel value in the image, then normalize each pixel with $x' = \frac{x-\mu}{\sigma}$
- **Square Root:** Per-pixel $x' = \sqrt{x}$ with input clipped at 0
- **Log1p:** Per-pixel $x' = \ln(1+x)$ with input clipped at 0
- **Percentile Clipping:** Compute the 1st and 99th percentile pixel values of the entire image, then clamp each pixel to within this range
- **Inter-Quartile Range, clipped (IQR-Clip):** A robust analog to mean/standard deviation normalization. After computing the first, second, and third quartiles of the input image, $x' = \frac{x-q_2}{q_3-q_1}$. Values more than 5 IQR from the median are clipped.
- **Inter-Quartile Range, soft clipped (IQR-Log):** After IQR normalization, apply a logarithmic nonlinearity to values above or below a certain deviation from the median (± 5 IQR in this experiment)

2.3 Training

All models were trained on NVIDIA A100 GPUs with 40GB of VRAM. With 512 x 512 input frames, we used a batch size of 16 in order to fit on a single A100 for every task/model combination. We used Focal Loss[11] for segmentation tasks, SmoothL1 loss for regression tasks. We used a OneCycle[17] learning rate schedule and a peak learning rate of 1e-3, which allowed us to train models to convergence in 25 epochs. We used an AdamW optimizer with weight decay of 0.1. For regression tasks, maximum learning rate was lowered to 1e-4. We expect that absolute performance of all models would improve with a longer training schedule, but the relative performance of the models appears to be stable with respect to training duration.

ViT Pre-Training We found that the ViT model did not perform well when trained with the same dataset and procedure as the other models. This is consistent with the findings of [4] and [18], which suggest that extensive pretraining or knowledge distillation are essential for ViT performance. We pre-trained all ViT backbones with a Masked Autoencoder (MAE) [6] on a similar but separately generated synthetic dataset, and then adapted to the downstream tasks with the same procedure as the other models.

2.4 Dataset

We generated synthetic scenes with corresponding ground truth information. With synthetic scenes, we can manipulate the statistics of the training and validation data to test the behavior of the networks under different conditions. We can also generate arbitrary amounts of training data to support training large models.

Our simulator injects targets and stars with random positions and velocities. We intentionally avoid simulating targets or stars based on catalogs, to prevent our models from learning about real constellations or the relative positions of GEO objects. We have found that generating artificially target-rich scenes (e.g. 200 targets per 512 x 512 pixel image) helps the models to learn more quickly and improves performance on real data. We image each scene with a different sensor whose parameters are randomly generated within specified bounds. These randomized sensors vary in ensquared energy, integration time, dark current, stray light, bad pixels, time-varying atmospheric seeing conditions, and structured noise levels.

For these experiments, we generated a dataset with 50,000 training scenes and 1,000 validation scenes. Each scene consists of a 512 x 512 image with associated ground truth information. Assuming a 1:32 resolution feature extractor, this corresponds to approximately 12M training patches for the segmentation task, and approximately 6M training chips for our chip extraction task before augmentation.

3. EXEMPLAR TASKS

We designed a set of tests to evaluate a network's ability to extract certain types of information from SDA imagery. We simulated datasets for each task as described in Section 2.4, with the simulation parameters in Appendix A. For each task, we designed a minimal network head to process the feature maps and produce the desired result. The task heads are intentionally simple in order to emphasize the effect of the backbone, and the same head architecture is used with each backbone. The backbones and heads are trained from random initialization for each task, with the exception of ViT.

3.1 Location Estimation from an Image Chip

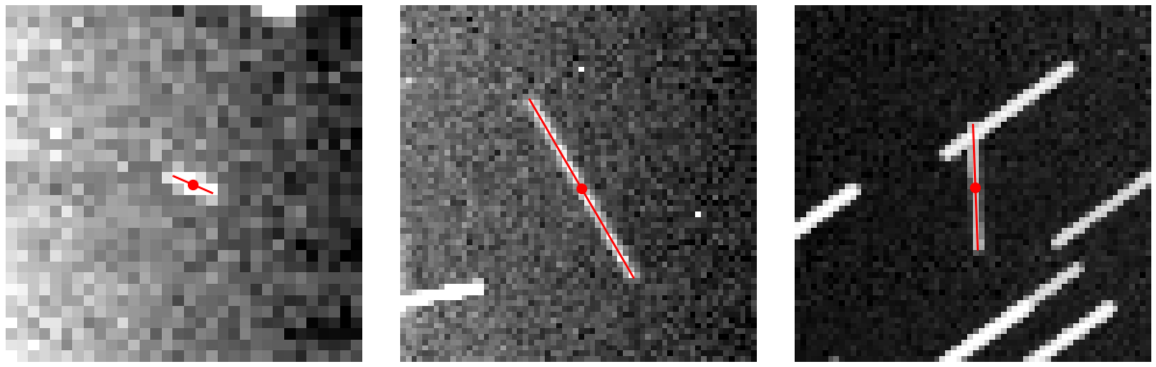

(a) Slow moving RSO (b) Fast moving RSO (c) Fast moving RSO

Fig. 3: Example chips with ground-truth annotation of centroid and velocity

3.1.1 Task Description

The network must estimate the centroid, or midpoint location, of a moving target from a small image chip around a synthetic detection. To emulate detections on the synthetic truth data, we select a random location along the length of the target streak, then perturb this location by up to ±3px in x and y. From an image centered on this detection location, the network must estimate the offset to the true centroid of the original target with sub-pixel precision. The centroid estimation head uses the feature map pixels corresponding to a 32x32 pixel chip in the input image.

This task is intended to measure a backbone's ability to extract fine spatial information.

Feature map resolution, or feature map downsampling factor, is an important attribute of a feature extraction backbone. Some architectures, such as FullRes, are designed to produce feature maps at pixel resolution with no downsampling. Others, such as ViT, produce features at a fixed downsampling factor (e.g 1:16). Hierarchical models such as ResNet produce a pyramid of features at different downsampling factors, which we fuse with a Feature Pyramid Network to produce feature maps at a number of resolutions (e.g. 1:4, 1:8, 1:16).

This test is especially sensitive to the method used to convert the model's coarse positional information to fine position estimates. We tested two variants of the centroiding head:

- **Bilinear:** Upsample feature maps from feature map resolution to pixel resolution using bilinear interpolation. Upsampled feature maps are flattened and passed to a linear layer which produces the x and y position estimates. Bilinear interpolation is the most common form of upsampling in FPN or UNet architectures.
- **PixelShuffle:** Identical to Bilinear, except that we replace bilinear interpolation with PixelShuffle [16], which should be capable of extracting high-spatial-frequency information from the feature maps

The measure of performance on this task is L_2 error between the predicted target centroid location and the actual target centroid location, in units of pixels.

3.1.2 Results

Table 2: Position estimation performance of different backbone architectures. Models with high downsample factors did not converge ("*") using bilinear interpolation. PixelShuffle extracts high-precision spatial information from models with large downsampling factor.

Architecture	Feature Map Resolution	Position Error Percentiles (Pixels L_2) Bilinear 50th	Bilinear 95th	PixelShuffle 50th	PixelShuffle 95th
Initial Detection Location		*2.43*	*4.11*	*2.43*	*4.11*
FullRes	1:1	**0.23**	**1.86**	**0.23**	**1.87**
U-Net	1:1	0.30	2.28	0.31	2.17
Swin+FPN	1:4	0.44	2.74	0.32	2.31
ResNet+FPN	1:4	0.39	2.29	0.28	2.23
XCiT	1:8	0.34	2.91	0.29	2.63
ViT	1:8	0.57	3.23	0.36	2.80
ResNet+FPN	1:4	0.39	2.29	0.28	2.23
ResNet+FPN	1:8	0.83	3.32	0.29	2.30
ResNet+FPN	1:16	*	*	0.32	2.45
XCiT	1:8	0.34	2.91	0.29	2.63
XCiT	1:16	*	*	0.43	3.19
ViT	1:8	0.57	3.23	0.36	2.80
ViT	1:16	*	*	0.69	3.76

Results are presented in Table 2. Models with more emphasis on high-resolution features perform exceptionally well on this task, producing median positional errors of less than 1/3rd of a pixel. With the default bilinear interpolation used by models like UNet or FPN, training did not converge for larger downsample models (1:16). By replacing bilinear interpolation with PixelShuffle, we can produce high-precision location estimates even from models with low spatial resolution. This implies that the backbones are capable of embedding this information, but care must be taken to allow a task head to access that information in the feature maps.

The positional errors are lower for ResNet+FPN at 1:16 downsample with PixelShuffle upsampling than for the ResNet+FPN 1:4 output. FPN uses bilinear interpolation to upsample and fuse feature maps to produce the 1:4 features, and it appears that PixelShuffle operation is better for preserving fine detail.

3.2 Low SNR Point Detection

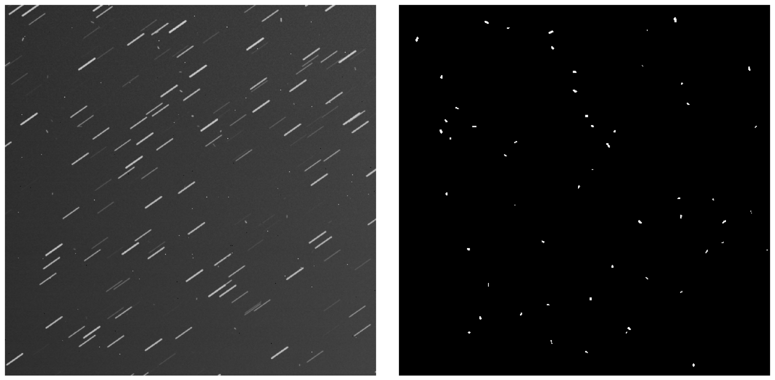

(a) Example Image (b) Expected Segmentation Mask

Fig. 4: Synthetic input image and ground-truth segmentation mask for Low SNR Point Detection

3.2.1 Task Description

The network must detect dim, point-like targets in imagery with stars, stray light, bad pixels, and noise. The scenes are reminiscent of data from fixed-staring GEO belt observations, with stars streaking across the image at a random angle and targets appearing as points or elongated blobs. We intend this task to assess a backbone's ability to extract subtle variations in intensity near the noise floor, with very little spatial support. There is no overlap between the simulated shapes of stars and the shapes of targets; the network can learn to identify targets by shape *a priori*.

From a theoretical perspective, optimal detection of these signatures in a Gaussian noise field would involve matched filtering to extract maximum signal from a small area of the image with minimal noise contribution. This task is more complex, since the networks must also reject non-Gaussian noise sources and confounding signals from stars or stray light.

We use a minimal segmentation head to produce a binary segmentation mask from the backbone features. Following from the results of the centroid estimation task, we use PixelShuffle to upsample the feature map to pixel resolution, followed by a linear layer and sigmoid activation to produce a binary segmentation mask.

The measures of performance on this task are Average Precision (AP)–precision averaged between multiple points on the precision-recall curve–and recall for targets with a peak SNR (pSNR) between 3 and 5. Targets with peak SNR below 5 are challenging for single-frame detection algorithms to detect with a low false detection rate. There are also targets below SNR of 3 and above SNR of 5 which contribute to the overall recall measurement.

3.2.2 Results

Results are presented in Table 3. The detection performance of the models is surprisingly clustered despite large variations in architecture, parameter count, and feature map resolution. This could imply that all models have found a solution which is a good compromise between recall and precision as prescribed by the Focal Loss training objective.

The noteworthy outliers are XCiT, which significantly outperforms the other models despite a relatively large downsampling factor (1:8), and ViT, which underperforms all other models while being the most computationally expensive by far. Large downsample factors significantly degrade the performance of the patch-based transformer algorithms (XCiT and ViT), but not the hierarchical features produced by ResNet+FPN.

Table 3: Detection performance of different model architectures for the Low SNR Point Detection task

	Feature Map Resolution	AP All	Recall pSNR 3-5	Recall All
FullRes	1:1	0.79	0.58	0.65
UNet	1:1	0.78	0.59	0.66
Swin+FPN	1:4	0.77	0.59	0.66
ResNet+FPN	1:4	0.81	0.62	0.69
XCiT	1:8	**0.85**	**0.67**	**0.74**
ViT	1:8	0.76	0.57	0.64
ResNet+FPN	1:4	0.81	0.62	0.69
ResNet+FPN	1:8	0.79	0.60	0.68
ResNet+FPN	1:16	0.78	0.59	0.66
ResNet+FPN	1:32	0.74	0.55	0.62
XCiT	1:8	**0.85**	**0.67**	**0.74**
XCiT	1:16	0.70	0.50	0.56
ViT	1:8	0.76	0.57	0.64
ViT	1:16	0.04	0.01	0.01
ViT	1:32	0.00	0.00	0.00

3.3 Anomalous Low SNR Streak Detection

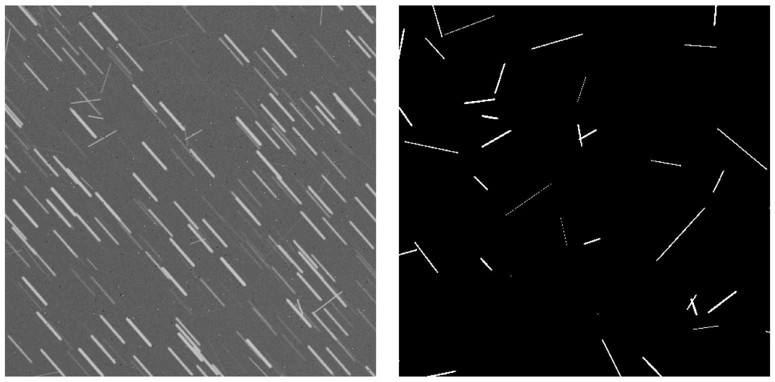

(a) Example Image (b) Expected Segmentation Mask

Fig. 5: Synthetic input image and ground truth segmentation mask for Anomalous Low SNR Streak Detection

3.3.1 Task Description

The network must detect dim streaking targets (moving >20px-frame) in imagery with stars, stray light, bad pixels, and noise. The scenes are analogous to detecting LEO or MEO objects from a fixed-staring sensor, with stars streaking at a random angle across the image. The model does not have access to a star catalog and must detect targets by their anomalous shape compared to the stars. However, the backbone can also integrate energy from multiple pixels to make a detection decision, which should improve detection performance relative to the point detection task if the model is able to utilize that additional signal.

The design of the segmentation head and measures of performance are identical to the Low SNR Point Detection task.

Table 4: Detection performance of different model architectures for the Anomalous Low SNR Streak Detection task

	Feature Map Resolution	AP All	Recall pSNR 3-5	Recall All
FullRes	1:1	0.57	0.38	0.33
UNet	1:1	0.86	0.75	0.71
Swin+FPN	1:4	0.92	0.85	0.82
ResNet+FPN	1:4	0.94	0.89	0.85
XCiT	1:8	**0.98**	**0.93**	**0.91**
ViT	1:8	0.95	0.89	0.86
ResNet+FPN	1:4	0.94	0.89	0.85
ResNet+FPN	1:8	0.93	0.85	0.82
ResNet+FPN	1:16	0.93	0.86	0.83
ResNet+FPN	1:32	0.87	0.79	0.75
XCiT	1:8	**0.98**	**0.93**	**0.91**
XCiT	1:16	0.93	0.86	0.84
ViT	1:8	0.95	0.89	0.86
ViT	1:16	0.88	0.77	0.75
ViT	1:32	0.15	0.04	0.06

3.3.2 Results

Results are presented in Table 4. For most architectures, the detection performance is higher in this experiment than for the point detection task despite similar pixelwise SNR. This suggests that these backbones are capable of using target energy spread over many pixels to make detection decisions. We also observe a much larger spread in detection performance between architectures on this task compared to point detection.

The detection performance of the FullRes model declined substantially. This is most likely due to the FullRes model's limited receptive field (15x15px), which makes it impossible for the model to gather context for anomalous streak detection. While it can likely "see" the targets, it cannot discriminate between a target and a star without context.

Conversely, the models with large receptive fields or global attention perform exceptionally well. Large feature map downsample factors were less detrimental on this task than for point detection, but still had a more significant effect on the XCiT and ViT models than ResNet. ViT performs well on this task, especially compared to its poor performance on localization and point detection tasks.

3.4 Bright Star Rejection

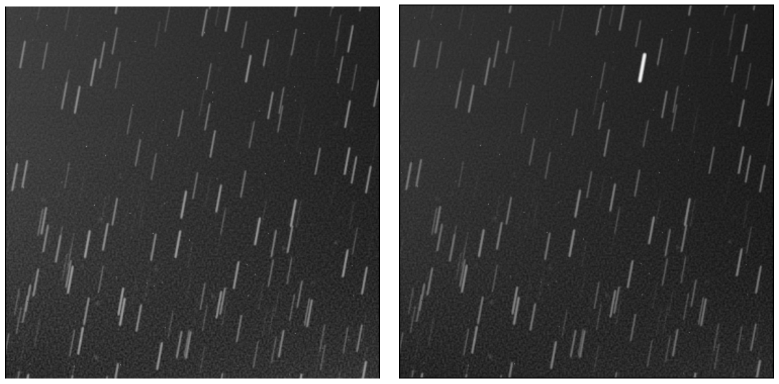

(a) Baseline Image (b) Bright Star Variant

Fig. 6: Synthetic input images for bright star rejection test. Detection behavior on both images should be identical if the model is robust.

3.4.1 Description

The Bright Star Rejection task is intended to test a model's robustness to high dynamic range data. We used models trained for the Low SNR Point Detection task and generated new validation scenes with the same simulator configuration. For each of these baseline scenes, we also generated a modified version with the intensity of the brightest star increased by a factor of 1000. Without retraining the networks, we measured their detection performance on the baseline and bright-star pairs. The measure of performance on this task is the change in AP and recall between the baseline and bright-star images.

3.4.2 Results

We tested with many architecture variants, and the results shown in Table 5 are exemplar of what we observed. We tested several models with ReLU or ReLU6 activation functions, and found that ReLU6 had no effect. We observed stronger effects from normalization layers and preprocessing. In Table 5 we show the performance of two variants of ResNet: one with BatchNorm and one with the GroupNorm layer that is popular in newer models, and XCiT which uses BatchNorm in the initial patch embedding stage and LayerNorm before transformer layers.

Table 5: Loss of detection performance in the vicinity of a bright star. Full results in Appendix B. Results with small loss of performance in bold.

Architecture	Normalization	Preprocessing	Baseline		Difference	
			AP	Recall	AP	Recall
ResNet	Batch	None	0.81	0.49	-24%	-6%
ResNet	Batch	Mean/Variance	0.80	0.48	-38%	-92%
ResNet	Batch	IQR-Log	0.79	0.46	**-0%**	**-0%**
ResNet	Batch	Log1p	0.81	0.50	**-0%**	**-0%**
ResNet	Group	None	0.71	0.27	-97%	-100%
ResNet	Group	Mean/Variance	0.81	0.51	-98%	-100%
ResNet	Group	IQR-Log	0.80	0.48	**-0%**	**-0%**
ResNet	Group	Log1p	0.81	0.50	**-0%**	**-0%**
XCiT	Batch/Layer	None	0.83	0.55	**-2%**	**-0%**
XCiT	Batch/Layer	Mean/Variance	0.84	0.58	-43%	-88%
XCiT	Batch/Layer	IQR-Log	0.84	0.57	**-0%**	**-0%**
XCiT	Batch/Layer	Log1p	0.85	0.58	**-1%**	**-0%**

Pairing GroupNorm with ResNet leads to a total degradation of detection performance when the bright star is injected. BatchNorm is more robust, but still suffers a moderate loss of detection precision without preprocessing. XCiT is robust to the presence of the bright star by default. This is noteworthy, because XCiT uses LayerNorm in the body of the network after patch embedding. LayerNorm and GroupNorm are closely related, and both apply inference-time mean/variance normalization, which appears to be the cause of the performance loss on the ResNet GroupNorm test. We suspect that the presence of large outliers, such as the bright star signal, corrupts these mean and variance estimates, leading to degraded performance. If that is the case, then why is XCiT unaffected?

To address this question, we note that the Mean/Variance preprocessing step severely degrades the performance of all models on the bright star test, including XCiT. Combining mean/variance preprocessing with BatchNorm is similar to applying the GroupNorm operation with a group size of 1 on the first layer of the network. It is possible that the impact of high dynamic range is greatest on the early layers of the neural network, and that models can compensate as long as the early layers are robust (BatchNorm, in the case of XCiT).

However, we also found that strong dynamic range compression, such as Log1p preprocessing, made all models robust to the bright star test and improved baseline detection performance. Log1p and IQR preprocessors allow us to use GroupNorm over BatchNorm, which can be essential to training large models [22]. While Log1p produced the best detection performance in all of our tests, we noted that IQR-Log stablized training and has standardizing behavior that may improve generalization to real datasets.

4. CONCLUSION

We demonstrated that modern neural network architectures for computer vision can provide excellent performance on Space Domain Awareness (SDA) image processing tasks. Some models require modification for robust performance in the presence of unresolved targets or high dynamic range. PixelShuffle upsampling layers unlock fine spatial details from large models, and strong dynamic range compression improved performance for all models tested. Cross-covariance image transformers (XCiT) were the strongest backbone architecture that we tested, with excellent detection performance, good preservation of spatial detail, stable training, and robustness to high dynamic range by default.

5. REFERENCES

[1] Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 34:20014–20027, 2021.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[6] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Andrew G Howard. Mo-bilenets: Efficient convolutional neural networks for mo-bile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[10] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[11] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[12] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[13] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[14] K. Merry, M. Dykstra, and B. Hacsi. Pixelwise image segmentation with convolutional neural networks for detection of resident space objects. In *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, September 2022.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international*

conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, pages 234–241. Springer, 2015.

[16] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[17] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

[18] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[19] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[20] Ross Wightman. Pytorch image models, 2019.

[21] D. Woodward, C. Manughian-Peter, T. Smith, and E. Davison. Pixelwise image segmentation with convolutional neural networks for detection of resident space objects. In *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, September 2021.

[22] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

6. COPYRIGHT NOTICE

A. SIMULATION PARAMETERS

	Distribution	Low_SNR_Detection	Low_SNR_Detection_Streaks
Number of Training Scenes	fixed	50000	50000
Number of Validation Scenes	fixed	1000	1000
Image Rows	fixed	512	512
Image Columns	fixed	512	512
Scene ensquared energy	uniform	(0.05, 0.60)	(0.05, 0.60)
Scene dark current	log-uniform	(10, 10^4)	(10, 10^4)
Scene stray light variance	log-uniform	(10, 10^4)	(10, 10^4)
Structured noise variance	log-uniform	(10, 10^3)	(10, 10^3)
Intensity of bad pixels	log-uniform	(10, 10^6)	(10, 10^6)
Number of bad pixels	log-uniform	(10, 10^3)	(10, 10^3)
Number of Targets	uniform	(50, 200)	(50, 200)
Target Intensities	log-uniform	(10, 10^4)	(10^2, 10^5)
Target Speeds	uniform	(1, 5)	(20, 100)
Target Directions (deg)	uniform	(0, 360)	(0, 360)
Number of Stars	uniform	(100, 1000)	(100, 1000)
Star Intensities	log-uniform	(10, 10^6)	(10, 10^6)
Scene Star Speed	uniform	(10, 50)	(10, 50)
Scene Star Direction (deg)	uniform	(0, 360)	(0, 360)

B. BRIGHT STAR REJECTION EXPERIMENTS

Architecture	Normalization	Preprocessing	Baseline		Bright Star		Difference	
			AP	Recall	AP	Recall	AP	Recall
ResNet	Batch	None	0.81	0.49	0.62	0.46	-0.19	-0.03
ResNet	Batch	Mean/Variance	0.80	0.48	0.50	0.04	-0.30	-0.44
ResNet	Batch	Percentile	0.80	0.49	0.71	0.48	-0.09	-0.01
ResNet	Batch	IQR-Clipped	0.79	0.48	0.71	0.47	=	-0.01
ResNet	Batch	IQR-Log	0.79	0.46	0.79	0.46	=	=
ResNet	Batch	Sqrt	0.00	0 00	0.00	0.00	=	=
ResNet	Batch	Log1p	0.81	0.50	0.81	0.50	=	=
ResNet	Group	None	0.71	0.27	0.01	0.00	-0.70	-0.27
ResNet	Group	Mean/Variance	0.81	0.51	0.02	0.00	-0.79	-0.51
ResNet	Group	Percent	0.81	0.51	0.44	0.21	-0.37	-0.30
ResNet	Group	IQR-Clipped	0.80	0.48	0.80	0.46	=	-0.02
ResNet	Group	IQR-Log	0.80	0.48	0.80	0.48	=	=
ResNet	Group	Sqrt	0.82	0.52	0.62	0.07	-0.20	-0.45
ResNet	Group	Log1p	0.81	0.50	0.81	0.50	=	=
XCiT	Layer	None	0.83	0.55	0.81	0.55	-0.02	=
XCiT	Layer	Mean/Variance	0.84	0.58	0.44	0.07	-0.40	-0.51
XCiT	Layer	Percent	0.84	0.57	0.82	0.57	-0.02	=
XCiT	Layer	IQR-Clipped	0.83	0.56	0.83	0.56	=	=
XCiT	Layer	IQR-Log	0.84	0.57	0.84	0.57	=	=
XCiT	Layer	Sqrt	0.83	0.54	0.82	0.54	-0.01	=
XCiT	Layer	Log1p	0.85	0.58	0.84	0.58	-0.01	=