# Automated Deep Learning Approach for Nucleus Segmentation in Fluorescence LEXY Videos

Bioinformatics Research Report

Chiem Bosboom

September 2024

Supervisor: Dr. Ihor Smal: Image Analysis, Smart Microscopy and AI

Second Examiner: Dr. Florian Berger: Theoretical Biophysics

Cell Biology, Neurobiology and Biophysics, Department of Biology, Faculty of Science, Utrecht University

Utrecht University

**Abstract**

Nucleus segmentation in fluorescence microscopy is crucial for studying nucleocytoplasmic transport, particularly with optogenetic systems like LEXY (light-inducible nuclear export). LEXY allows precise, reversible control of nuclear protein export via irradiation, but videos of cells expressing LEXY present significant segmentation challenges. As irradiation progresses, the nuclear-cytoplasmic contrast diminishes, leading to frames where the nucleus becomes difficult or impossible to distinguish from the cytosol. To address these challenges, an automated deep learning approach is proposed that integrates temporal information to enhance the accuracy of segmentation in these LEXY fluorescence videos. Using pre-training on the large, high-quality Human Protein Atlas (HPA) dataset, I evaluated U-Net and two additional architectures, Track-Net and Siam-Net, which incorporate the previous frame mask and a reference frame and mask, respectively. Given the heterogeneity of the LEXY dataset, which contains only 10 videos, synthetic videos were generated from HPA data designed to mimic LEXY conditions. In these synthetic sequences, Track-Net and Siam-Net exhibited significant error propagation over time, resulting in poor segmentation performance. In contrast, the base U-Net model produced more consistent, though still suboptimal, predictions. Testing the base U-Net on the actual LEXY dataset showed that pre-training with HPA improved segmentation, but still achieving a disappointing Dice score of 0.55. It is clear that more training data and exploration of alternative architectures is needed to fully address the segmentation challenges posed by LEXY. The code is available on GitHub.

**Plain Language Summary**

In this study, I focused on improving the automatic segmentation of cell nuclei in microscopy videos featuring the LEXY (light-inducible nuclear export) system. LEXY makes it possible to control the movement of proteins out of the cell nucleus with light. However, as proteins move out of the nucleus, the contrast between the nucleus and the surrounding cell becomes less clear, making it difficult to accurately segment the nucleus in certain frames. To solve this problem, I used a type of artificial intelligence (AI) called deep learning. Specifically, I trained a model called U-Net, along with two different versions called Track-Net and Siam-Net, to help improve the segmentation of nuclei in microscopy videos. I used a large dataset of images from the Human Protein Atlas (HPA) to generate synthetic videos that mimic the challenging conditions in the LEXY videos. Although these models showed promise, both Track-Net and Siam-Net struggled to maintain accuracy over time, as small errors in earlier frames led to larger mistakes in later ones. The basic U-Net model was more consistent but still did not perform as well as I had hoped. In the end, pre-training the models on the larger HPA dataset did improve their ability to segment nuclei, but more data and different approaches are needed to fully solve the challenges posed by LEXY videos.

# 1 Introduction

Nucleocytoplasmic transport is a fundamental cellular process by which molecules cross the double membrane nuclear envelope to shuttle between the nucleus and the cytosol. This transport is facilitated by nuclear pore complexes (NPCs), which make the nuclear envelope selectively permeable to macromolecules. The Karyopherin-$\beta$ (Kap) family of nuclear transport receptors mediates most nucleocytoplasmic transport by recognizing macromolecules with nuclear localisation or export signals (NLSs or NESs), binding to them, and transporting them across an NPC. The localisation of these macromolecules is critical for various cellular processes, including cell cycle regulation and signal transduction [1].

Taking advantage of this mechanism, the light-inducible nuclear export system (LEXY) offers a powerful optogenetic tool for controlling protein localisation. LEXY consists of the LOV2 core and J$\alpha$ helix of *Avena sativa phototropin-1*, which has been engineered to include an NES (Fig. 1). Under normal conditions, the J$\alpha$ helix is tightly packed against the LOV2 core, preventing the recognition of NES. However, upon exposure to blue light, the J$\alpha$ helix unfolds, allowing the NES to be recognised and exported from the nucleus. With the addition of a constitutive NLS, LEXY can be genetically fused to any protein of interest, offering reversible nuclear export using light [2].

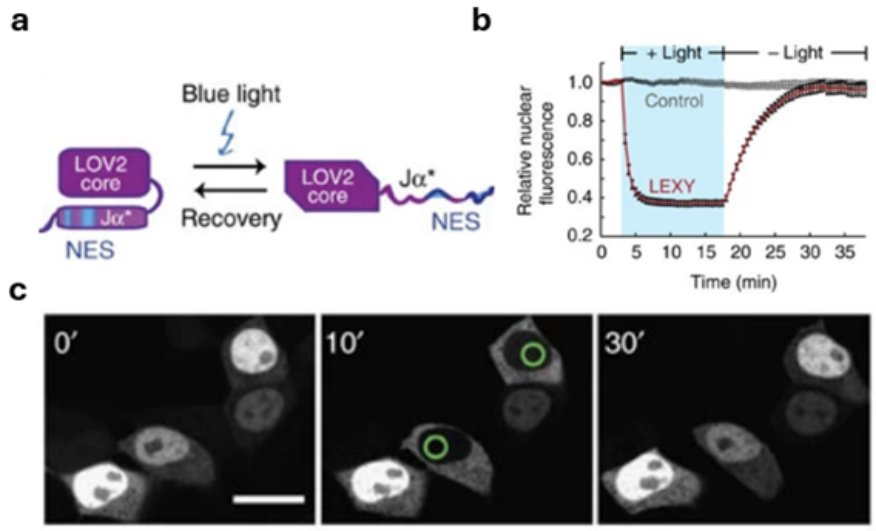



**Figure 1: Characterisation of LEXY.** (**a**) Graphic illustrating LEXY functionality. Exposure to blue light uncages NES from the LOV2 core, leading to the export of proteins from the nucleus. (**b**) Relative nuclear fluorescence over time in cells expressing NLS-mCherry-LEXY and control. Cells were kept in darkness for 3 minutes, followed by 15 minutes of blue light irradiation and a subsequent 20 minute dark recovery period (mean ± s.e.m., n = 22 cells from 3 independent experiments). (**c**) Fluorescence images of cells expressing NLS-mCherry-LEXY. Green circles indicate areas irradiated with a blue laser for 10 minutes followed by a 20 minute dark recovery period. Scale bar: 20 $\mu$m. Adapted from [2].

LEXY variants, such as iLEXY$i$ and iLEXY$s$, have been developed with point mutations (V416I and V416L) that increase cytoplasm localisation both before and after light irradiation, improving nuclear depletion [3, 4]. In contrast, a light-inducible nuclear localisation signal (LINuS) has been developed to import proteins in response to light instead of exporting proteins [5]. Interestingly, the light-activated nuclear shuttle (LANS) and the light-inducible nuclear exporter (LINX), similar to LINuS and LEXY, respectively, were independently developed around the same time with slight differences in implementation [6, 7].

Despite the versatility of these systems, accurate quantification of nucleocytoplasmic transport remains a challenge. One of the key obstacles is the segmentation of the nucleus and cytosol in fluorescence microscopy videos. Accurate segmentation of these cellular compartments is essential for measuring intensity differences. Although manual segmentation offers higher accuracy, automatic segmentation is more efficient and reproducible, especially in large-scale studies or real-time applications. Quick, automatic segmentation may even allow for dynamic control of protein localisation by adjusting protein export rates through changes in laser intensity.

Existing nucleus segmentation models, such as StarDist [8], Cellpose [9], and NucleAIzer [10], have demonstrated success across various modalities, including fluorescently labeled cell cultures and H&E-stained tissues. However, most of the datasets used in these models rely on Hoechst or DAPI staining, which distinctly visualises the nucleus [11]. In contrast, much of generated fluorescent data do not have a nuclear marker because it occupies a valuable fluorescent channel. Although the nucleus is usually visible in LEXY videos, some frames have poor contrast between the nucleus and cytosol, which complicates segmentation. Moreover, the LEXY dataset used here is small, heterogeneous, and features weak nuclear ground truth masks, making it difficult to effectively train on.

To address these challenges, I propose a novel approach that leverages pre-training on a large, high-quality dataset, the Human Protein Atlas (HPA), which contains thousands of fluorescent images similar to LEXY images. I use the DAPI channel from these HPA images to create weak nuclear ground truth masks using Cellpose. My objective is to improve the precision of segmentation by integrating temporal information. Here, I compare U-Net [12] against what will be referred to as Track-Net [13], and Siam-Net [14], which incorporate the previous frame mask and a reference frame, respectively. Finally, I will explore the impact of different loss functions, data augmentation, and transfer learning to optimise segmentation performance.

# 2 Methods

## 2.1 Datasets

### 2.1.1 LEXY

The LEXY dataset consists of 10 fluorescence microscopy videos, each capturing a single U-2 OS osteosarcoma epithelial-like cell stably expressing NLS-mCherry-LEXY from Addgene. Microscopy was performed using a wide-field DIC Phase Contrast FL 'PEX'-scope, with a 40X objective and a 1.5X tube lens, resulting in a 60X total magnification. Each video contains between 21 and 41 frames, starting with blue light irradiation, and ending in a dark recovery phase. Photostimulation was carried out using a 470 nm LED and a Polygon 400 digital mirror device from Mightex.

Initially, the nucleus appears relatively bright, but as irradiation goes on, the nuclear brightness decreases while the cytosol becomes slightly brighter. Once irradiation stops, this process reverses: the nucleus becomes brighter, and the cytosol dims. However, as the difference in fluorescence between the nucleus and cytosol reduces, it becomes increasingly difficult to accurately segment the nucleus, eventually reaching a point where the nucleus is no longer visible.

In Addition, the LEXY dataset shows significant variation in brightness, contrast, and noise across the videos, with bright spots in the background further complicating segmentation. To improve consistency across videos, the sequences were normalised between 0 and 1 based on background intensity. The data was split into 10 folds, with each fold corresponding to one of the 10 cell videos for cross-validation.

Ground truth masks were generated by labeling the nucleus with SiR-DNA [15]. Cells were incubated with 500 nM SiR-DNA for 30 minutes prior to imaging, and the dye was kept in the media throughout the imaging process. SiR-DNA brightly labels the nucleus, allowing for segmentation via thresholding.

### 2.1.2 HPA

To increase training data and improve segmentation performance, the HPA was leveraged, which contains confocal microscopy images (63x magnification) of immunofluorescence-labeled human proteins. The dataset includes 79,408 images representing 13,145 proteins. For this study, a subset of images was selected based on subcellular location. Only proteins localised in the cytosol, the nucleoplasm, or both were chosen, excluding images with other structures or organelles.

The final dataset contains 5,936 images with cytosol localisation, 12,457 images with nucleoplasm localisation, and 5,119 images with both cytosol and nucleoplasm localisation, totaling 23,512 images. However, due to unreliability in labeling, many images labeled as nucleoplasm-only or cytosol-only actually visualise both compartments. The

data were split into training (80%), validation (10%), and test (10%) sets. All images include a DAPI channel, which was used to generate ground truth masks for the nucleus.

**Mask Generation**

To create ground truth nucleus masks from the DAPI channel, I compared segmentation results from three methods: thresholding, Cellpose, and StarDist (Fig. 2). While StarDist effectively generated uniform, convex shapes and separated close neighbors, it struggled to consistently produce accurate edges. Thresholding provided more accurate segmentation in complex images with varying nuclei sizes but made errors when artifacts or close neighbors were present. Cellpose ultimately showed the best balance between accuracy and consistency, likely thanks to its ability to estimate nuclear size automatically. Some HPA images contain densely packed, small cells, which are not suitable for pre-training. However, Cellpose was able to generate small enough predictions to allow effective filtering of these images. After removing small objects, any images without nucleus masks were excluded from the dataset.

**Image Cropping and Preprocessing**

Once nucleus masks were made, each image was cropped to a quarter of its original size to better match the resolution of LEXY images. Excluding border touching nuclei, crops were centered around each individual nucleus. However, in order to properly represent the LEXY dataset, the images were first filtered on cell density (8 or fewer nuclei per image, excluding border-touching nuclei). Lastly, small objects were removed from the cropped masks to ignore any barely visible nuclei on the edges, resulting in a final dataset of 27,154 training images, 3,192 test images, and 3,372 validation images. While being stricter on cell density (e.g., isolating single cells) was considered, I opted for this density threshold to maintain diversity in the dataset. A balance was struck to avoid reducing the training data too much, which would limit the model's generalisability.

**Synthetic Sequence Generation**

To further mimic the LEXY videos, I also created synthetic sequences from these cropped images. Images were fist filtered further, selecting only those with no border-touching nuclei and where nuclear and cytosolic intensities were relatively similar. This ensured that intensity adjustments and transformations applied during synthetic sequence generation would result in realistic looking frames.

Images were progressively rotated, stretched, and sheared to simulate cellular translocation. This adds more difficulty to the segmentation task, as simply copy-pasting the previous frame mask is no longer viable. Additionally, the nuclear intensity was adjusted gradually across frames to simulate the intensity changes seen in LEXY videos. A total of 153 test sequences and 97 validation sequences were generated, with each sequence

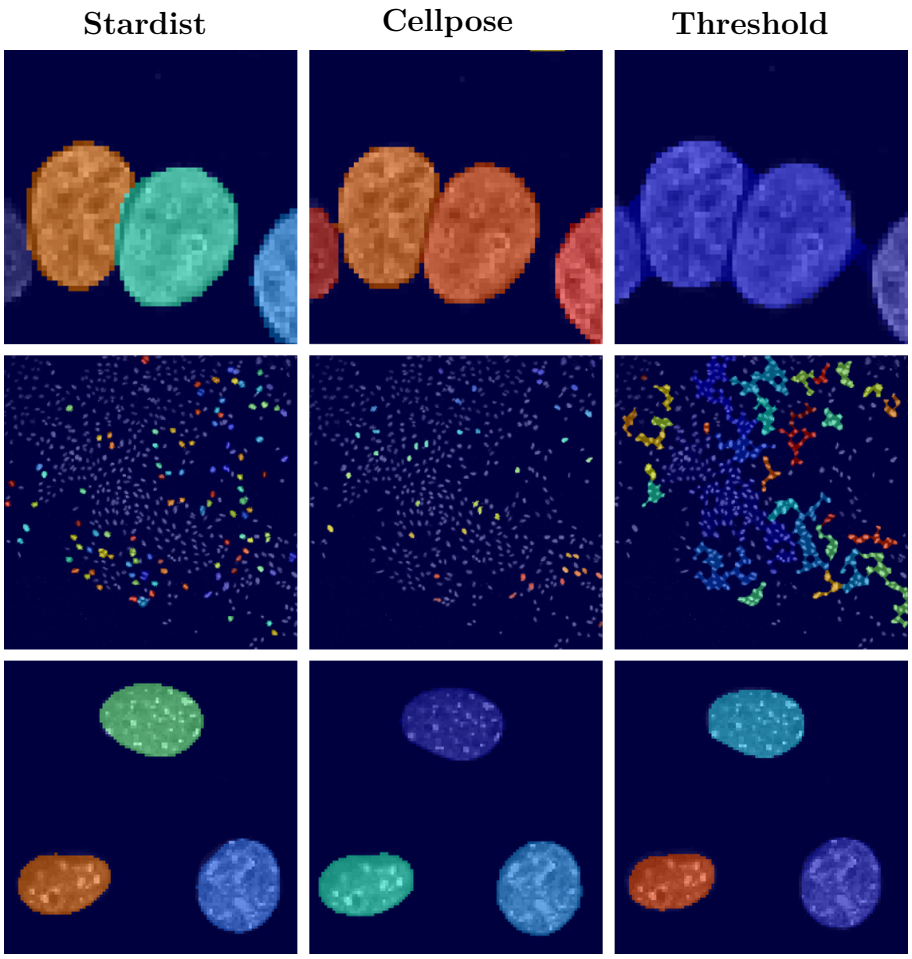



**Figure 2: Comparing Stardist, Cellpose, and Thresholding for Nucleas Segmentation.** Images show examples of DAPI-stained nuclei from HPA with corresponding predictions. The top and bottom row zooms show that Cellpose is more accurate in comparison to Stardist and thresholding. The middle row shows a full size example with small nuclei. Both Stardist and Cellpose make small enough segmentations to easily filter these images out.

comprising 30 frames.

## 2.2 Model Architectures

### 2.2.1 Convolutional Neural Networks

Fully connected neural networks are impractical for images because the number of parameters grows exponentially as input dimensions increase. Additionally, fully connected layers do not account for the spatial relationships between neighboring pixels in an image. Convolutional neural networks (CNNs) address these limitations by using convolutional layers specifically designed for image data [16].

In a convolutional layer, a kernel (small matrix of learnable weights) slides across the input image to compute so-called feature maps (Fig. 3). Kernels are usually of size 3x3 or have odd and equal dimensions to keep symmetry. For each pixel in the input image, the

kernel multiplies the pixel and its surrounding neighbors by their respective weights. The sum of these weighted values and a bias is then passed through an activation function like rectified linear unit (ReLU), which sets all negative values to zero. This process is repeated across the entire image, resulting in a feature map of the same dimensions as the input image.

The stride controls the step size of the kernel as it moves across the image. A stride of 1 will move the kernel one pixel at a time, while a stride of 2 skips every other pixel, reducing the resolution by half. In addition, padding will maintain the original input size after convolution by adding a border of zeros. Without padding, dimensions will always be reduced by one because the kernel does not have enough surrounding pixels to perform a full operation on the border pixels.

Each convolutional layer usually has multiple feature maps, with each map representing a unique aspect of the image learned by the network. The number of feature maps, or channels determines how many kernels go over the input. Each feature map has its own set of learnable kernel weights and a learnable bias.

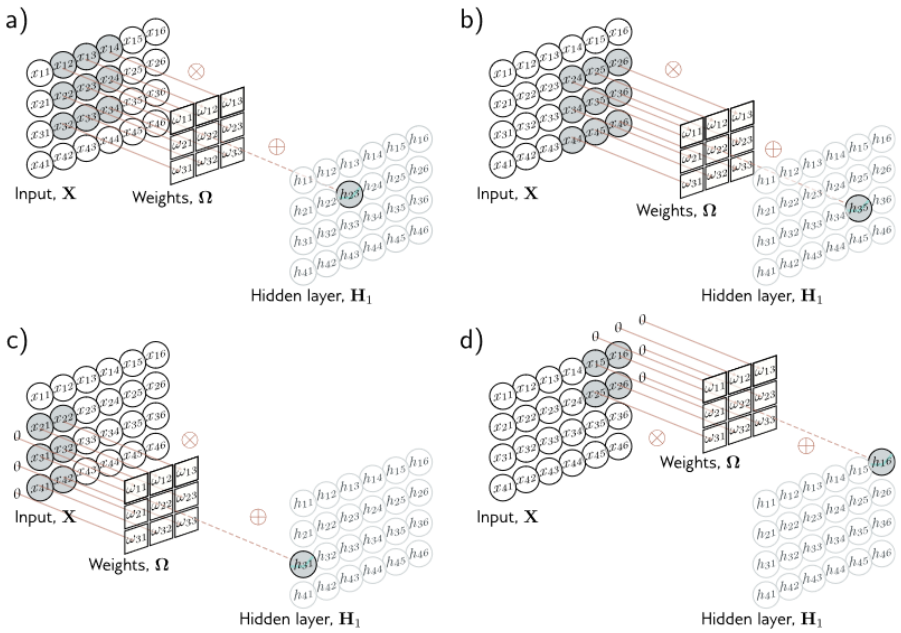



**Figure 3: Convolutional layers.** The images show how a 3x3 kernel moves across the input image to create a feature map. The input, weights and hidden layer are represented by $x$, $\omega$, and $h$, respectively. Shaded areas are included in the operation. With zero-padding and a stride of 1, each pixel and its surrounding neighbors are multiplied by their respective weights. The sum is then added to a bias and passed through an activation function. From [16].

### 2.2.2 U-Net

U-Net [12] is a CNN which has seen widespread success in medical imaging thanks to its precise segmentation with relatively limited training data. This is especially helpful when

working with medical datasets like LEXY, where annotated data is often limited. U-Net has a flexible design, which has seen many different iterations, using ideas like adversarial training, densely connected layers, residual connections, and attention mechanisms [17].

U-Net consists of two main parts: an encoder (downsampling path) and a decoder (upsampling path), connected via a bottleneck (Fig. 4). The encoder progressively reduces the spatial dimensions of the input image while increasing the number of channels to capture increasingly abstract features. Conversely, the decoder progressively increases the spatial dimensions and reduces the number of channels back to the image's original size.

The encoding path consists of a series of 3x3 convolutional layers, each followed by a ReLU activation and 2x2 max-pooling. Max-pooling reduces the spatial resolution by half by selecting the maximum value from every 2x2 block. Starting with 64 channels, the number of channels is doubled after each downsampling step, up to a depth of 4. At the bottom of U-Net, the bottleneck connects the encoder and decoder. The decoding path mirrors the encoder but uses up-convolution, effectively the opposite of a regular convolution, to double the spatial resolution. In addition, at each level, the upsampled feature maps are concatenated with the corresponding feature maps from the encoder via skip connections. These skip connections enable U-Net to combine high-resolution information from the encoder with the abstracted features in the decoder, improving localisation and segmentation of small structures in the image.

In this study, the base U-Net is used with some small modifications. The original has no padding, leading to reductions in resolution with each convolution. Here, padding of 1 is used in all convolutional layers to keep consistency between feature maps at different levels. The final output is reduced to a single channel for binary segmentation.

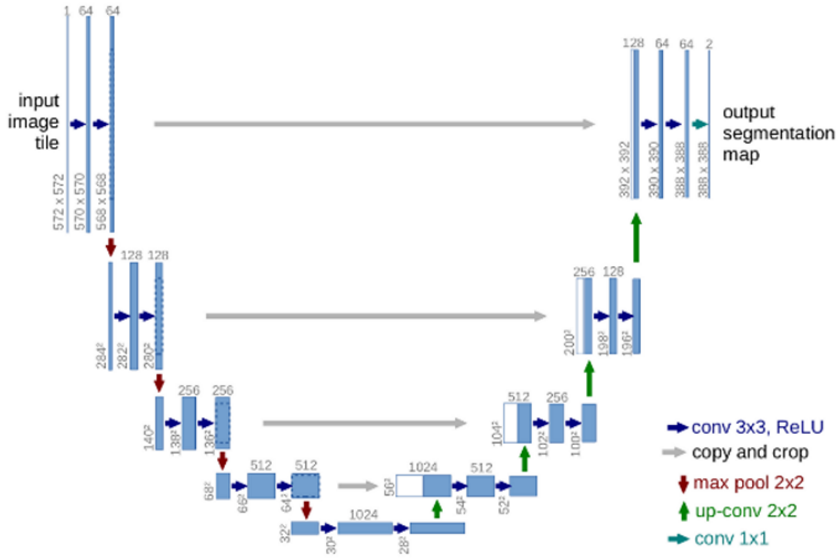



**Figure 4: U-Net.** Each blue box represents a multi-channel feature map, with the number of channels indicated above the box. The dimensions are shown at the bottom left corner. White boxes indicate duplicated feature maps, and the arrows illustrate the various operations. From [12].

### 2.2.3 Track-Net

Track-Net is based on the idea of [13] to incorporate temporal information for video segmentation. Track-Net uses the same U-net architecture described above, but includes an additional input channel for the previous frame mask. This allows the model to propagate and adjust previous predictions, which should be especially useful in difficult frames. The key benefit of this model is the ability to train on single images by generating synthetic estimates for the previous frame's mask. Here, I used a random combination of shearing, stretching, and rotation to get a Dice score of 0.80 compared to the ground truth. As the difference between the previous frame mask and the current frame increases, the model is expected to rely less on the previous mask for making new predictions. However, in some LEXY frames, accurate segmentation depends entirely on the previous frame. A balance has to be struck to ensure the model does not simply copy and paste, but also is not completely independent, allowing it to use the previous frame when segmentation is challenging.

### 2.2.4 Siam-Net

Siam-Net follows the same approach but introduces a second encoder branch to process a reference frame and its corresponding mask based on [14]. This Siamese network architecture share parameters between encoders. The outputs of both encoders are concatenated at the bottleneck of U-Net, with no additional skip connections between the second encoder and the decoder. The idea of this design is to reduce error propagation

by providing the model with a reliable reference frame. For training, synthetic estimates of the reference frame and mask were made by shifting the previous frame mask by a multitude of 2.

## 2.3 Transfer Learning

Although base U-Net is highly effective for many segmentation tasks, its performance can be further improved through transfer learning [18, 19]. This approach involves replacing the original encoder with a pre-trained model, typically trained on a large and diverse dataset, allowing the model to benefit from previously learned feature representations. As a result, transfer learning leads to faster convergence, improved generalisation, and enhanced accuracy. In this study, I implemented two widely used middle-ground classification encoders: VGG16 [20] and ResNet34 [21], both pre-trained on the ImageNet dataset. ImageNet contained 1.28 million hand-annotated natural training images with 1,000 object categories, used for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) from 2012-2017 [22]. Since then, ImageNet has grown to over 14 million images.

### 2.3.1 VGG

The VGG network shares much of its structure with U-Net's encoder (Fig. 5). VGG16 consists of five convolutional blocks, with the last three blocks containing three convolutional layers instead of two. Three fully connected layers then reduce the feature map to 1000 classes for ImageNet classification. There are a number of different ways VGG can be put into U-Net [23, 24, 25, 26, 27, 28]. In my implementation, the fully connected layers of VGG16 were replaced with U-Net's bottleneck. Given that VGG16 has a depth of five convolutional blocks (in contrast to U-Net's original depth of four), I extended the U-Net decoder with an additional block and skip connection. Lastly, VGG16 was designed for three channel inputs. This was changed to either one channel for U-Net or two for Track-Net and Siam-Net. Note that Siam-Net uses two shared VGG16 encoders, the outputs from which are concatenated at the the bottleneck.


**Figure 5: VGG19.** Each color represents a different spatial dimension. The text inside each box indicates the kernel size and the number of channels. White boxes indicate fully connected layers. From [21].

### 2.3.2 ResNet

ResNet introduces residual connections that allow feature maps to skip past layers, reducing the vanishing gradient problem and facilitating the training of deeper networks [21]. ResNet34 consists of residual blocks, each containing two 3x3 convolutional layers followed by batch normalisation and ReLU, with residual connections after each block. Like VGG, there are a number of different ways to implement ResNet into U-Net [29, 30, 31, 32]. In this study, I modified the architecture by removing the final fully connected layer and adjusting the network to a depth of five by removing the first max-pooling step. This adjustment was necessary because of the initial 7x7 convolution, which has a stride of 2, immediately reducing the input resolution. Because of this, the highest resolution skip connection was excluded, as it would map directly to the original image. Lastly, ResNet34 was adapted to accept one or two channel inputs depending on the model.

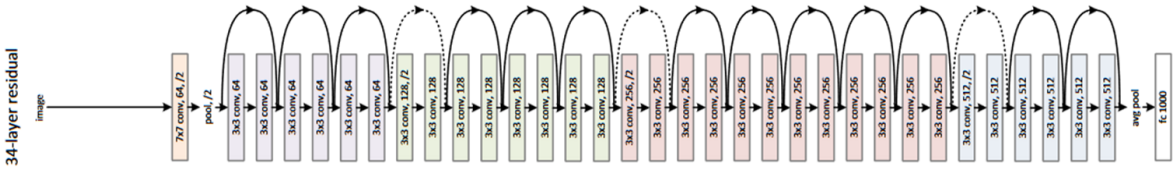



**Figure 6: ResNet34.** Each color represents a different spatial dimension. The text inside each box indicates the kernel size, the number of channels, and stride. Residual connections are indicated with arrows, where dotted lines indicate a reduction in spatial dimensions. White boxes indicate fully connected layers. From [21].

## 2.4 Training

The models were trained using the Adam optimiser, with an initial learning rate of 0.0001. For the HPA dataset, U-net and Track-Net were trained for 100 epochs with a batch size of 64, while Siam-Net was trained for 50 epochs with a batch size of 32. For the LEXY dataset, all models were trained via cross-validation for 10 epochs with a batch size of 32.

### 2.4.1 Loss Functions

In this study, three standard segmentation loss functions were compared [33]. Firstly, Binary Cross-Entropy (BCE) loss, common for binary classification tasks. It measures the pixel-wise difference between the predicted probability and the true label. BCE loss is defined as:

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(w \cdot y_i \log(p_i) + (1-y_i)\log(1-p_i)\right) \tag{1}$$

Where N is the total number of pixels in the image, $y_i$ is the the true label for pixel $i$ (1 for nucleus, 0 for background), $p_i$ is the predicted probability of pixel $i$ being part of the nucleus, and $w$ is a weighting factor for the nucleus class. Here, $w$ is set to 2.

Secondly, Dice loss, derived from the Dice coefficient, a measure of overlap between the predicted and ground truth segmentation. A higher Dice coefficient corresponds to a better overlap, so Dice loss subtracts this value from 1 to make it a loss function. Dice loss is defined as:

$$\text{Dice Loss} = 1 - \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{2}$$

Where TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives, and FN is the number of false negatives.

Lastly, a combined loss of BCE loss and Dice loss, which can be defined as:

$$\text{Combined Loss} = \text{BCE Loss} + \beta \cdot \text{Dice Loss} \tag{3}$$

where $\beta$ is a weighting factor that balances the contribution of Dice loss relative to BCE loss. Here, $\beta$ is set to 1.

### 2.4.2 Performance Metrics

To evaluate model performance, multiple metrics were used, including Dice coefficient (F1 score), sensitivity (recall), specificity, and Hausdorff distance (HD) [34]. As explained above, the Dice coefficient measures overlap between ground truth and prediction. For binary classification tasks, dice is equal to F1 score and is defined as:

$$\text{Dice} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{4}$$

Sensitivity, also known as recall, measures the proportion of actual positives correctly identified by the model. High Sensitivity indicates fewer false negatives. Sensitivity is defined as:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5}$$

Conversely, Specificity measures the proportion of actual negatives that were correctly identified. High Specificity indicates fewer false positives. Specificity is defined as:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{6}$$

Lastly, HD measures how far two sets of points are from each other. It quantifies the maximum distance between the boundary pixels of the predicted and ground truth

segmentation. A lower HD indicates better alignment of boundaries. HD is defined as:

$$H(A, B) = \max\left(\max_{a \in A} \min_{b \in B} d(a, b), \max_{b \in B} \min_{a \in A} d(a, b)\right) \tag{7}$$

Where $A$ is the set of boundary pixels in the ground truth segmentation, $B$ is the set of boundary pixels in the predicted segmentation, $\max_{a \in A} \min_{b \in B} d(a, b)$ is the maximum of the minimum Euclidean distance between each point in $A$ to the closest point in $B$, and $\max_{b \in B} \min_{a \in A} d(a, b)$ is the maximum of the minimum distance between each point in $B$ to the closest point in $A$.

### 2.4.3 Data Augmentation

Data augmentation was used during training to improve model generalisation and prevent overfitting. I applied rotations and flips randomly as standard augmentation techniques. For the LEXY dataset, I also used perspective changes and random cropping as extra augmentations for the smaller dataset [35, 36].

# 3 Results

Given the small size and variability of the LEXY dataset, the preliminary results were highly influenced by the choice of individual cells for testing. Even with cross-validation across all 10 videos, results varied significantly between training runs. This inconsistency makes it difficult to find a model that performs well with LEXY data. To address this, results from the larger HPA dataset and synthetic videos designed to closely resemble LEXY data are compared. The goal is to find a model that generalises well to future LEXY datasets, rather than optimising for the current LEXY videos alone.

Three model architectures are compared: U-Net, Track-Net, and Siam-Net. U-Net serves as the baseline model. Track-Net extends U-Net by adding the previous frame mask as additional input. Siam-Net builds on Track-Net by adding a second encoder that shares parameters with the main encoder, using a reference frame and mask as input. For training, estimates of the previous frame mask and reference frame are generated where necessary. During testing, only the first frame uses these estimates, subsequent frames use the model's actual predictions.

Each model is evaluated across nine configurations, combining three loss functions: Dice loss, BCE loss, and a combined Dice-BCE loss, with three encoder choices: the base U-Net encoder, VGG16 pre-trained on ImageNet, and ResNet34 pre-trained on ImageNet. More details in 2.

## 3.1 U-Net Performance

The base U-Net architecture showed consistent performance improvements when using pre-trained encoders (Tab. 1). Although differences across loss functions were minimal, the use of pre-trained encoders consistently led to small reductions in Dice and HD, indicating improved segmentation. Specifically, ResNet34 and VGG16 averaged a Dice score of 0.92 compared to 0.91 for the base U-Net encoder, and ResNet34 averaged a HD of 14.2, compared to 14.7 for VGG16, and 16.3 for U-Net. However, as seen in the segmented images, ResNet34 struggles in certain cases, while the other models perform comparably well (Supplementary Fig. 5.1), suggesting VGG16 may be the better option.

**Table 1:** Performance metrics on HPA for U-Net with different pre-trained encoders and loss functions.

| Encoder, Loss | **Sensitivity** | **Specificity** | **Dice Coefficient** | **Hausdorff Distance** |
|---|---|---|---|---|
| U-Net, BCE | 0.9232 ± 0.1214 | 0.9864 ± 0.0148 | 0.9096 ± 0.1088 | 16.5128 ± 21.7158 |
| U-Net, Combined | 0.9143 ± 0.1193 | 0.9882 ± 0.0148 | 0.9108 ± 0.1059 | 16.2027 ± 21.1397 |
| U-Net, Dice | 0.9228 ± 0.1113 | 0.9877 ± 0.0152 | 0.9141 ± 0.1035 | 16.2237 ± 21.6353 |
| ResNet34, BCE | 0.9300 ± 0.0980 | 0.9887 ± 0.0140 | 0.9228 ± 0.0912 | 14.2986 ± 20.2864 |
| ResNet34, Combined | **0.9354 ± 0.0905** | 0.9876 ± 0.0156 | 0.9230 ± 0.0907 | 14.2814 ± 20.2880 |
| ResNet34, Dice | 0.9283 ± 0.0933 | **0.9897 ± 0.0140** | **0.9256 ± 0.0889** | **14.0519 ± 20.1748** |
| VGG16, BCE | 0.9329 ± 0.0963 | 0.9879 ± 0.0150 | 0.9218 ± 0.0915 | 14.5921 ± 20.2503 |
| VGG16, Combined | 0.9286 ± 0.0986 | 0.9886 ± 0.0144 | 0.9216 ± 0.0913 | 14.7409 ± 20.1757 |
| VGG16, Dice | 0.9288 ± 0.1026 | 0.9893 ± 0.0136 | 0.9238 ± 0.0932 | 14.6830 ± 20.4226 |

Fluctuating validation loss curves show models overfitting the training data when using BCE and combined loss (Fig. 7). In contrast, Dice loss converged more smoothly and consistently, suggesting that the learning rate may have been too high for BCE and combined loss. Had the models using these loss functions been properly fitted, they may have performed slightly better. For this reason, no definitive conclusions can be made about loss functions. Nevertheless, pre-trained encoders consistently showed faster convergence, further pointing to their effectiveness.

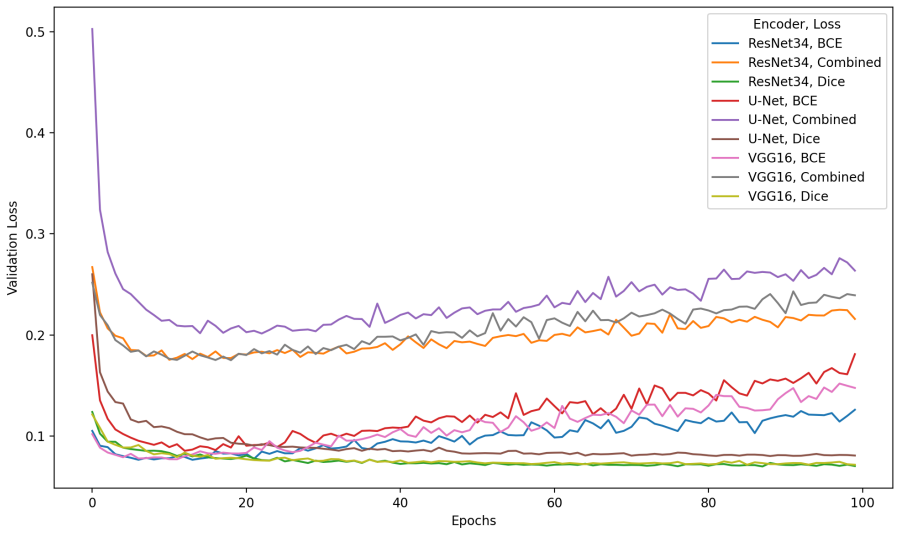



**Figure 7:** Validation loss over epochs during HPA training for U-Net with different encoders and loss functions.

On synthetic HPA sequences, pre-trained encoders also showed improved performance (Tab. 2. ResNet34 averaged a Dice score of 0.93 and a HD of 10.2, while VGG16 averaged 0.92 and 11.2, and U-Net averaged 0.91 and 13.3. Although, these models performed well on frames with clearly visible nuclei, segmentation quality decreased on difficult frames like frame 24, where no nucleus was visible (Supplementary Fig. 5.1). However,

performance on these difficult frames was higher than expected, predictions being quite reasonable. This indicates that the synthetic HPA sequences might not accurately reflect the complexity of LEXY data, potentially due to unintentional patterns emerging from sequence generation.

**Table 2:** Performance metrics on synthetic HPA sequence for U-Net with different pre-trained encoders and loss functions.

| Encoder, Loss | **Sensitivity** | **Specificity** | **Dice Coefficient** | **Hausdorff Distance** |
|---|---|---|---|---|
| U-Net, BCE | 0.9566 ± 0.0781 | 0.9851 ± 0.0101 | 0.9066 ± 0.0681 | 13.9791 ± 16.4550 |
| U-Net, Combined | 0.9487 ± 0.0804 | 0.9867 ± 0.0096 | 0.9088 ± 0.0666 | 13.1475 ± 16.1666 |
| U-Net, Dice | 0.9554 ± 0.0709 | 0.9860 ± 0.0102 | 0.9099 ± 0.0629 | 12.6184 ± 15.7096 |
| ResNet34, BCE | 0.9615 ± 0.0515 | 0.9883 ± 0.0094 | 0.9238 ± 0.0493 | 11.1171 ± 15.8435 |
| ResNet34, Combined | 0.9631 ± 0.0508 | 0.9878 ± 0.0096 | 0.9230 ± 0.0535 | 10.7142 ± 15.8175 |
| ResNet34, Dice | 0.9613 ± 0.0506 | **0.9898 ± 0.0080** | **0.9300 ± 0.0476** | **8.8603 ± 12.8678** |
| VGG16, BCE | **0.9659 ± 0.0582** | 0.9875 ± 0.0092 | 0.9219 ± 0.0533 | 11.4278 ± 15.5707 |
| VGG16, Combined | 0.9600 ± 0.0676 | 0.9881 ± 0.0091 | 0.9202 ± 0.0649 | 12.0678 ± 16.1616 |
| VGG16, Dice | 0.9611 ± 0.0588 | 0.9883 ± 0.0084 | 0.9237 ± 0.0505 | 10.0338 ± 12.7525 |

## 3.2 Track-Net Performance

Track-Net showed even greater fluctuations in validation loss during training, possibly due to the increased complexity of the architecture (Fig. 8). However, there were no meaningful differences in performance between loss functions. Interestingly, while pre-trained encoders continued to accelerate convergence, they also led to more pronounced fluctuations and overfitting. This is reflected in the performance metrics: ResNet34 and VGG16 averaged Dice scores of 0.93, whereas U-Net averaged a Dice score of 0.95 (Tab. 3). U-Net also outperformed in HD, with 6.6 compared to 7.1 for ResNet34 and 7.2 for VGG16.

**Table 3:** Performance metrics on HPA for Track-Net with different pre-trained encoders and loss function.

| Encoder, Loss | **Sensitivity** | **Specificity** | **Dice Coefficient** | **Hausdorff Distance** |
|---|---|---|---|---|
| U-Net, BCE | **0.9573 ± 0.0413** | 0.9899 ± 0.0070 | 0.9447 ± 0.0368 | 6.7847 ± 11.4461 |
| U-Net, Combined | 0.9536 ± 0.0382 | **0.9917 ± 0.0061** | **0.9489 ± 0.0326** | 6.6144 ± 11.4615 |
| U-Net, Dice | 0.9544 ± 0.0408 | 0.9902 ± 0.0066 | 0.9443 ± 0.0353 | **6.2698 ± 10.0041** |
| ResNet34, BCE | 0.9414 ± 0.0444 | 0.9866 ± 0.0091 | 0.9272 ± 0.0398 | 7.3469 ± 10.6458 |
| ResNet34, Combined | 0.9371 ± 0.0452 | 0.9871 ± 0.0088 | 0.9263 ± 0.0400 | 7.1907 ± 9.9604 |
| ResNet34, Dice | 0.9372 ± 0.0482 | 0.9884 ± 0.0083 | 0.9299 ± 0.0422 | 6.8093 ± 9.4360 |
| VGG16, BCE | 0.9404 ± 0.0455 | 0.9871 ± 0.0088 | 0.9277 ± 0.0413 | 7.0554 ± 9.9262 |
| VGG16, Combined | 0.9350 ± 0.0466 | 0.9882 ± 0.0079 | 0.9282 ± 0.0406 | 7.2863 ± 10.4417 |
| VGG16, Dice | 0.9277 ± 0.0485 | 0.9889 ± 0.0083 | 0.9268 ± 0.0418 | 7.2057 ± 10.0549 |

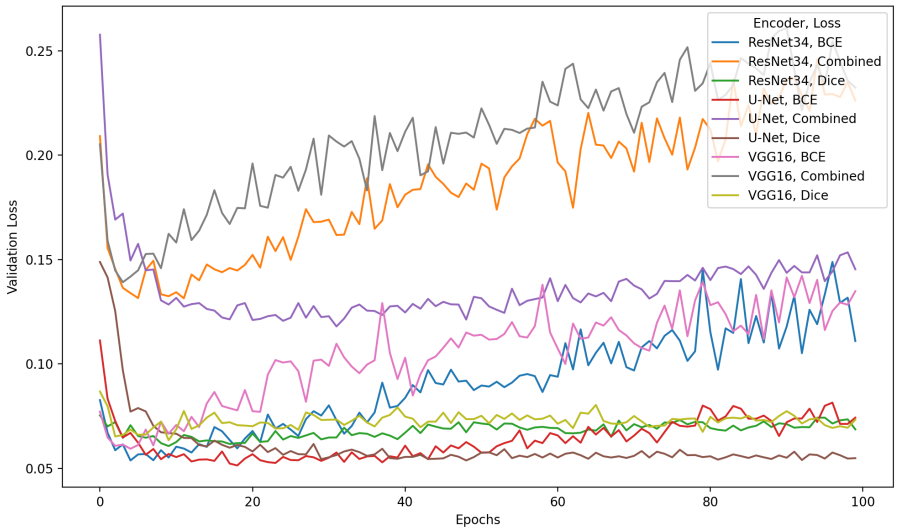



**Figure 8:** Validation loss over epochs during HPA training for Track-Net with different encoders and loss functions.

On synthetic sequences, Track-Net showed an over-reliance on previous frame masks, leading to error propagation and diminished performance in later frames (Supplementary Fig. 5.2). Track-Net performed notably worse compared to U-Net (Tab. 4). For instance, using the base U-Net encoder and Dice loss, Track-Net averaged a Dice score of 0.83 and a HD of 9.1, whereas U-Net averaged a Dice score of 0.91 and an HD of 12.6. This suggests that while Track-Net can effectively use previous frame masks to improve performance in initial frames, it struggles to maintain accuracy over longer sequences.

**Table 4:** Performance metrics on synthetic HPA sequence for Track-Net with different pretrained encoders and loss functions.

| Encoder, Loss | **Sensitivity** | **Specificity** | **Dice Coefficient** | **Hausdorff Distance** |
|---|---|---|---|---|
| U-Net, BCE | **0.8953 ± 0.0793** | 0.9895 ± 0.0066 | **0.8950 ± 0.0575** | **6.5923 ± 6.4393** |
| U-Net, Combined | 0.8039 ± 0.1358 | 0.9930 ± 0.0052 | 0.8473 ± 0.1105 | 9.3108 ± 9.1135 |
| U-Net, Dice | 0.7961 ± 0.1434 | 0.9905 ± 0.0082 | 0.8337 ± 0.1099 | 9.1203 ± 7.2301 |
| ResNet34, BCE | 0.7796 ± 0.1589 | 0.9801 ± 0.0164 | 0.7914 ± 0.1423 | 12.6414 ± 11.4448 |
| ResNet34, Combined | 0.7635 ± 0.1452 | 0.9905 ± 0.0074 | 0.8175 ± 0.1123 | 10.8219 ± 8.2853 |
| ResNet34, Dice | 0.5967 ± 0.2429 | 0.9916 ± 0.0052 | 0.6799 ± 0.2034 | 16.7896 ± 13.0124 |
| VGG16, BCE | 0.7780 ± 0.1312 | 0.9871 ± 0.0099 | 0.8162 ± 0.1074 | 10.1240 ± 7.3509 |
| VGG16, Combined | 0.7307 ± 0.1659 | 0.9899 ± 0.0072 | 0.7892 ± 0.1306 | 11.9615 ± 9.9958 |
| VGG16, Dice | 0.4942 ± 0.2710 | **0.9941 ± 0.0039** | 0.5877 ± 0.2455 | 19.1363 ± 13.2567 |

## 3.3 Siam-Net Performance

Siam-Net showed improvements over both U-Net and Track-Net on single images. With the reduced batch size (32) and training limited to 50 epochs, there is noticeably less noise and overfitting in validation curves (Fig. 9). However, this improved generalisation

may also be attributed to the model's architecture. Interestingly, pre-trained encoders now performed even worse compared to the base U-Net encoder, failing to pass the loss curves of U-Net throughout training. This is evident from the results: U-Net averaged a Dice score of 0.98 and a HD of 3.6, while ResNet34 averaged 0.96 and 4.7, and VGG16 averaged 0.95 and 5.1 (Tab. 5). Meanwhile, differences between loss functions stayed minimal.

**Table 5:** Performance metrics on HPA for Siam-Net with different pre-trained encoders and loss functions.

| Encoder, Loss | **Sensitivity** | **Specificity** | **Dice Coefficient** | **Hausdorff Distance** |
|---|---|---|---|---|
| U-Net, BCE | **0.9939 ± 0.0089** | 0.9968 ± 0.0023 | 0.9862 ± 0.0101 | **3.4814 ± 8.4316** |
| U-Net, Combined | 0.9925 ± 0.0090 | 0.9971 ± 0.0023 | **0.9864 ± 0.0097** | 3.5515 ± 8.7424 |
| U-net, Dice | 0.9821 ± 0.0164 | **0.9972 ± 0.0018** | 0.9814 ± 0.0115 | 3.6108 ± 8.1893 |
| ResNet34, BCE | 0.9686 ± 0.0264 | 0.9948 ± 0.0042 | 0.9668 ± 0.0221 | 4.7191 ± 8.2913 |
| ResNet34, Combined | 0.9679 ± 0.0276 | 0.9941 ± 0.0047 | 0.9644 ± 0.0242 | 4.6941 ± 7.6456 |
| ResNet34, Dice | 0.9617 ± 0.0293 | 0.9937 ± 0.0051 | 0.9599 ± 0.0259 | 4.7699 ± 7.5448 |
| VGG16, BCE | 0.9645 ± 0.0322 | 0.9928 ± 0.0053 | 0.9584 ± 0.0272 | 4.9413 ± 7.9227 |
| VGG16, Combined | 0.9602 ± 0.0327 | 0.9935 ± 0.0050 | 0.9585 ± 0.0272 | 4.8366 ± 7.4341 |
| VGG16, Dice | 0.9501 ± 0.0419 | 0.9912 ± 0.0060 | 0.9454 ± 0.0352 | 5.5841 ± 7.9092 |

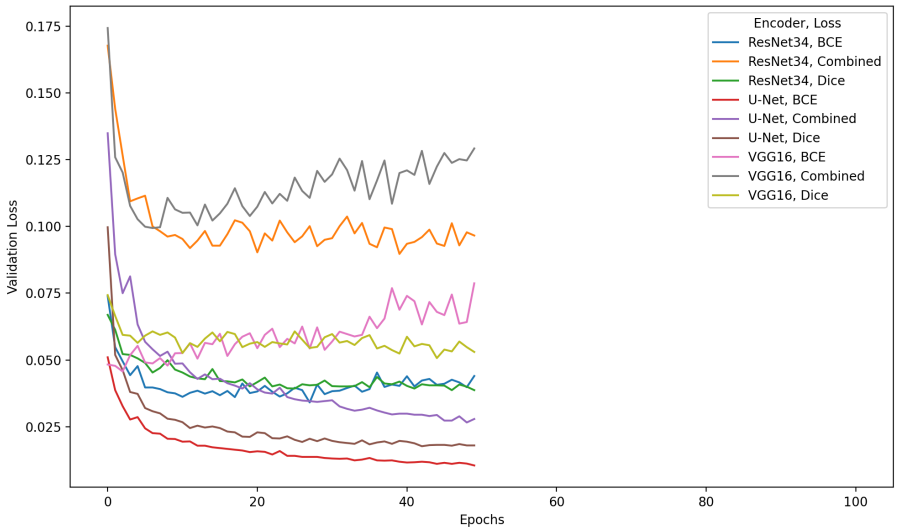



**Figure 9:** Validation loss over epochs during HPA training for Siam-Net with different encoders and loss functions.

However, on synthetic HPA sequences, SiamNet's performance was massively reduced (Tab. 6). Using the base U-Net encoder and Dice loss, Siam-Net averaged a Dice score of 0.57 and a HD of 28.6, performing worse than Track-Net. Therefore, introducing a reference frame and mask has not helped error propagation, but made it worse (Supplementary Fig. 5.3).

**Table 6:** Performance metrics on synthetic HPA sequence for Siam-Net with different pre-trained encoders and loss functions.

| Encoder, Loss | **Sensitivity** | **Specificity** | **Dice Coefficient** | **Hausdorff Distance** |
|---|---|---|---|---|
| U-Net, BCE | 0.6026 ± 0.1912 | 0.9712 ± 0.0265 | 0.6181 ± 0.1863 | 23.3789 ± 16.5542 |
| U-Net, Combined | 0.6946 ± 0.1680 | 0.9715 ± 0.0210 | 0.6873 ± 0.1634 | 19.2247 ± 14.0371 |
| U-Net, Dice | 0.5587 ± 0.1945 | 0.9692 ± 0.0218 | 0.5662 ± 0.1953 | 28.5864 ± 19.5297 |
| ResNet34, BCE | 0.5792 ± 0.2175 | 0.9916 ± 0.0087 | 0.6396 ± 0.2084 | 19.9790 ± 16.1983 |
| ResNet34, Combined | 0.5786 ± 0.2044 | 0.9928 ± 0.0068 | 0.6501 ± 0.1932 | 21.2723 ± 15.5539 |
| ResNet34, Dice | 0.6966 ± 0.1860 | **0.9955 ± 0.0038** | 0.7616 ± 0.1657 | 14.6803 ± 12.9708 |
| VGG16, BCE | 0.5863 ± 0.1966 | 0.9879 ± 0.0138 | 0.6365 ± 0.1929 | 22.2883 ± 16.7585 |
| VGG16, Combined | 0.5608 ± 0.2180 | 0.9853 ± 0.0137 | 0.6013 ± 0.2195 | 21.6943 ± 16.9973 |
| VGG16, Dice | **0.7345 ± 0.1718** | 0.9915 ± 0.0066 | **0.7815 ± 0.1520** | **10.9854 ± 9.9078** |

## 3.4 LEXY Performance

Given the poor performance of Track-Net and Siam-Net on the synthetic HPA sequences, I evaluated the performance of U-Net on the LEXY dataset. Dice loss was chosen for this evaluation based on previous training results, though any loss function would have sufficed given the minor differences. The effect of pre-training on HPA data, using a pre-trained VGG16 encoder, and combining both approaches were compared against base U-Net. VGG16 was chosen for its performance compared to ResNet34, which made poor predictions on certain images (Supplementary Fig. 5.1).

Cross-validation was performed across all ten LEXY videos twice, and results were averaged (Tab. 7). Results showed significant benefits from pre-training on HPA data, with a Dice score of 0.55 and a Hausdorff distance of 23.5, compared to no pre-training, which resulted in a Dice score of 0.34 and a HD of 31.4. VGG16 provided a smaller improvement over the base U-Net, with a Dice score of 0.43 and a HD of 24.4. However, combining both VGG16 and HPA pre-training did not offer additional benefits, with a Dice score of 0.54 and a HD of 23.2. Despite these improvements, results on the LEXY dataset remained unsatisfactory, making huge errors (Supplementary Fig. 5.4).

**Table 7:** Performance metrics on LEXY for U-Net with Dice loss.

| Model | **Sensitivity** | **Specificity** | **Dice Coefficient** | **Hausdorff Distance** |
|---|---|---|---|---|
| U-Net | 0.5670 ± 0.0663 | 0.9110 ± 0.0142 | 0.3437 ± 0.0523 | 31.3939 ± 2.5414 |
| VGG16 U-Net | **0.7376 ± 0.0241** | 0.9194 ± 0.0023 | 0.4349 ± 0.0203 | 24.3971 ± 1.2805 |
| HPA pre-trained U-Net | 0.5833 ± 0.2585 | **0.9747 ± 0.0098** | **0.5474 ± 0.2450** | 23.4787 ± 12.2138 |
| HPA pre-trained VGG16 U-Net | 0.5802 ± 0.2341 | 0.9721 ± 0.0110 | 0.5430 ± 0.2120 | **23.1530 ± 10.4057** |

# 4 Discussion and Conclusion

This study presents a new approach to improving nucleus segmentation in fluorescence microscopy by using pre-training on the large, high-quality HPA dataset. Two models that extend the base U-Net with temporal information were tested on synthetic HPA videos designed to mimic LEXY dynamics. Track-Net includes the previous frame mask, while Siam-Net goes further by adding a second encoder for the reference frame and mask. These models aim to enhance U-Net's single frame processing by adding information from earlier frames.

The results highlight a clear trade-off between the benefits of using temporal information and the challenge of error propagation. Siam-Net performed best on the HPA validation set, with Track-Net close behind. However, this success did not carry over to synthetic videos, where relying on accurate previous predictions caused errors to accumulate, leading to a big drop in performance. It is clear that these models need to handle noisy or inaccurate predictions better. Training with more unreliable previous and reference frame estimates might help, but there could also be deeper issues with the model architectures. A variation of Siam-Net, where the main encoder processes the current frame, and the second encoder uses only the previous frame mask, might reduce the impact of the previous frame by limiting its influence to a single point of connection at the bottleneck.

Pre-trained encoders, particularly VGG16, improved segmentation accuracy for base U-Net. However, the same pre-trained encoders worsened performance for Track-Net and Siam-Net due to overfitting. Interestingly, different loss functions, including Dice, BCE, and their combination, had little impact on segmentation performance across all models, suggesting that the model architecture may matter more than loss functions in this case.

Although base U-Net performed worse on HPA data, it produced more stable results on synthetic sequences compared to Track-Net and Siam-Net. However, its lack of temporal context limited its overall performance. On LEXY videos, performance dropped substantially due to the challenges of training on small, heterogeneous datasets. Pre-training on HPA data and using a pre-trained VGG16 encoder helped improve results, but combining them did not offer further benefits. As a result, a successful architecture for accurately segmenting nuclei in LEXY videos remains elusive.

Looking forward, there are several promising directions for future research. Regularisation techniques, such as dropout and batch normalisation, could help reduce overfitting and fluctuations in loss curves. Expanding data augmentation past the spatial transformations used in this study (rotation, shearing, random cropping, and flipping) to include methods like elastic deformations, scaling, sharpness, blur, noise, brightness, and contrast adjustments could increase the variation in training data. However, the vast number of possible augmentation combinations can sometimes hinder training, so tools

like AutoAugment could help find the most effective options [36, 37]. Additionally, experimenting with alternative loss functions, such as HD [38] and L1/L2 regularisation, may further improve segmentation performance [39].

To expand the LEXY dataset without the need for manual labeling, self-supervised or semi-supervised learning approaches offer great potential [40]. Furthermore, Generative Adversarial Networks (GANs) could generate synthetic LEXY images from nucleus segmentations, adding more annotated data to improve training [41].

In terms of model architecture, exploring recurrent neural networks (RNNs), such as long short-term memory (LSTM) networks, could enhance video segmentation by learning patterns across frames [42, 43]. This approach could be particularly effective for LEXY data, where pixel intensity changes indicate the boundary between nucleus and cytosol. Finally, while this study focused on nucleus segmentation, extending these methods to cytosol segmentation is a natural next step.

Footnote: For this project, I used ChatGPT from OpenAI to review code, generate ideas, and provide feedback on my writing style.

# References


[1] Casey E. Wing, Ho Yee Joyce Fung, and Yuh Min Chook. "Karyopherin-mediated nucleocytoplasmic transport". In: *Nat Rev Mol Cell Biol* 23.5 (May 2022), pp. 307–328. ISSN: 1471-0072. DOI: 10.1038/s41580-021-00446-7. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10101760/ (visited on 09/02/2024).

[2] Dominik Niopek et al. "Optogenetic control of nuclear protein export". en. In: *Nat Commun* 7.1 (Feb. 2016). Publisher: Nature Publishing Group, p. 10624. ISSN: 2041-1723. DOI: 10.1038/ncomms10624. URL: https://www.nature.com/articles/ncomms10624 (visited on 06/20/2024).

[3] Giada Forlani et al. "Analysis of Slow-Cycling Variants of the Light-Inducible Nuclear Protein Export System LEXY in Mammalian Cells". In: *ACS Synth. Biol.* 11.10 (Oct. 2022). Publisher: American Chemical Society, pp. 3529–3533. DOI: 10.1021/acssynbio.2c00232. URL: https://doi.org/10.1021/acssynbio.2c00232 (visited on 07/01/2024).

[4] Anna C. Kögler et al. "Extremely rapid and reversible optogenetic perturbation of nuclear proteins in living embryos". In: *Dev Cell* 56.16 (Aug. 2021), 2348–2363.e8. ISSN: 1534-5807. DOI: 10.1016/j.devcel.2021.07.011. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8387026/ (visited on 09/03/2024).

[5] Dominik Niopek et al. "Engineering light-inducible nuclear localization signals for precise spatiotemporal control of protein dynamics in living cells". en. In: *Nat Commun* 5.1 (July 2014). Publisher: Nature Publishing Group, p. 4404. ISSN: 2041-1723. DOI: 10.1038/ncomms5404. URL: https://www.nature.com/articles/ncomms5404 (visited on 06/20/2024).

[6] Hayretin Yumerefendi et al. "Control of Protein Activity and Cell Fate Specification via Light-Mediated Nuclear Translocation". In: *PLoS One* 10.6 (June 2015), e0128443. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0128443. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4471001/ (visited on 09/03/2024).

[7] Hayretin Yumerefendi et al. "Light-induced nuclear export reveals rapid dynamics of epigenetic modifications". In: *Nat Chem Biol* 12.6 (June 2016), pp. 399–401. ISSN: 1552-4450. DOI: 10.1038/nchembio.2068. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4888063/ (visited on 07/01/2024).

[8] Uwe Schmidt et al. "Cell Detection with Star-Convex Polygons". en. In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*. Ed. by Alejandro F. Frangi et al. Cham: Springer International Publishing, 2018, pp. 265–273. ISBN: 978-3-030-00934-2. DOI: 10.1007/978-3-030-00934-2_30.

[9] Carsen Stringer et al. “Cellpose: a generalist algorithm for cellular segmentation”. en. In: *Nat Methods* 18.1 (Jan. 2021). Publisher: Nature Publishing Group, pp. 100–106. ISSN: 1548-7105. DOI: 10.1038/s41592-020-01018-x. URL: https://www.nature.com/articles/s41592-020-01018-x (visited on 09/08/2024).

[10] Reka Hollandi et al. “nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer”. English. In: *cels* 10.5 (May 2020). Publisher: Elsevier, 453–458.e6. ISSN: 2405-4712, 2405-4720. DOI: 10.1016/j.cels.2020.04.003. URL: https://www.cell.com/cell-systems/abstract/S2405-4712(20)30117-4 (visited on 09/08/2024).

[11] Reka Hollandi et al. “Nucleus segmentation: towards automated solutions”. In: *Trends in Cell Biology* 32.4 (Apr. 2022), pp. 295–310. ISSN: 0962-8924. DOI: 10.1016/j.tcb.2021.12.004. URL: https://www.sciencedirect.com/science/article/pii/S0962892421002518 (visited on 09/08/2024).

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. en. arXiv:1505.04597 [cs]. May 2015. URL: http://arxiv.org/abs/1505.04597 (visited on 09/04/2024).

[13] Anna Khoreva et al. *Learning Video Object Segmentation from Static Images*. en. arXiv:1612.02646 [cs]. Dec. 2016. URL: http://arxiv.org/abs/1612.02646 (visited on 09/08/2024).

[14] Seoung Wug Oh et al. “Fast Video Object Segmentation by Reference-Guided Mask Propagation”. en. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, June 2018, pp. 7376–7385. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00770. URL: https://ieeexplore.ieee.org/document/8578868/ (visited on 09/08/2024).

[15] Gražvydas Lukinavičius et al. “SiR–Hoechst is a far-red DNA stain for live-cell nanoscopy”. en. In: *Nat Commun* 6.1 (Oct. 2015). Publisher: Nature Publishing Group, p. 8497. ISSN: 2041-1723. DOI: 10.1038/ncomms9497. URL: https://www.nature.com/articles/ncomms9497 (visited on 09/18/2024).

[16] Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, 2023. URL: http://udlbook.com.

[17] Nahian Siddique et al. “U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications”. In: *IEEE Access* 9 (2021). Conference Name: IEEE Access, pp. 82031–82057. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3086020. URL: https://ieeexplore.ieee.org/abstract/document/9446143 (visited on 09/15/2024).

[18] Mohammed Hakim Bendiabdallah and Nesma Settouti. "A comparison of U-net backbone architectures for the automatic white blood cells segmentation". en. In: *WAS Science Nature (WASSN) ISSN: 2766-7715* 4.1 (Sept. 2021). Number: 1. ISSN: 2766-7715. URL: http://worldascience.org/journals/index.php/wassn/article/view/24 (visited on 09/04/2024).

[19] Padmavathi Kora et al. "Transfer learning techniques for medical image analysis: A review". In: *Biocybernetics and Biomedical Engineering* 42.1 (Jan. 2022), pp. 79–107. ISSN: 0208-5216. DOI: 10.1016/j.bbe.2021.11.004. URL: https://www.sciencedirect.com/science/article/pii/S0208521621001297 (visited on 09/11/2024).

[20] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. en. arXiv:1409.1556 [cs]. Apr. 2015. URL: http://arxiv.org/abs/1409.1556 (visited on 09/04/2024).

[21] Kaiming He et al. *Deep Residual Learning for Image Recognition*. en. arXiv:1512.03385 [cs]. Dec. 2015. URL: http://arxiv.org/abs/1512.03385 (visited on 09/11/2024).

[22] Olga Russakovsky et al. *ImageNet Large Scale Visual Recognition Challenge*. arXiv:1409.0575 [cs]. Jan. 2015. DOI: 10.48550/arXiv.1409.0575. URL: http://arxiv.org/abs/1409.0575 (visited on 09/17/2024).

[23] Jesline Daniel et al. "VGG-UNet/VGG-SegNet Supported Automatic Segmentation of Endoplasmic Reticulum Network in Fluorescence Microscopy Images". en. In: *Scanning* 2022.1 (2022). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2022/7733860, p. 7733860. ISSN: 1932-8745. DOI: 10.1155/2022/7733860. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/7733860 (visited on 09/04/2024).

[24] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. en. arXiv:1511.00561 [cs]. Oct. 2016. URL: http://arxiv.org/abs/1511.00561 (visited on 09/04/2024).

[25] Seifedine Kadry et al. "Automated segmentation of leukocyte from hematological images—a study using various CNN schemes". en. In: *J Supercomput* 78.5 (Apr. 2022), pp. 6974–6994. ISSN: 1573-0484. DOI: 10.1007/s11227-021-04125-4. URL: https://doi.org/10.1007/s11227-021-04125-4 (visited on 09/04/2024).

[26] Vladimir Iglovikov and Alexey Shvets. *TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation*. en. arXiv:1801.05746 [cs]. Jan. 2018. URL: http://arxiv.org/abs/1801.05746 (visited on 09/04/2024).

[27] Maayan Frid-Adar et al. “Improving the Segmentation of Anatomical Structures in Chest Radiographs Using U-Net with an ImageNet Pre-trained Encoder”. en. In: *Image Analysis for Moving Organ, Breast, and Thoracic Images*. Ed. by Danail Stoyanov et al. Cham: Springer International Publishing, 2018, pp. 159–168. ISBN: 978-3-030-00946-5. DOI: 10.1007/978-3-030-00946-5_17.

[28] Ali Nawaz et al. “VGG-UNET for Brain Tumor Segmentation and Ensemble Model for Survival Prediction”. en. In: *2021 International Conference on Robotics and Automation in Industry (ICRAI)*. Rawalpindi, Pakistan: IEEE, Oct. 2021, pp. 1–6. ISBN: 978-1-66542-343-4. DOI: 10.1109/ICRAI54018.2021.9651367. URL: https://ieeexplore.ieee.org/document/9651367/ (visited on 09/04/2024).

[29] Jean Le’Clerc Arrastia et al. “Deeply Supervised UNet for Semantic Segmentation to Assist Dermatopathological Assessment of Basal Cell Carcinoma”. en. In: *Journal of Imaging* 7.4 (Apr. 2021). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 71. ISSN: 2313-433X. DOI: 10.3390/jimaging7040071. URL: https://www.mdpi.com/2313-433X/7/4/71 (visited on 09/11/2024).

[30] Jason Charng et al. “Deep learning segmentation of hyperautofluorescent fleck lesions in Stargardt disease”. In: *Scientific Reports* 10 (Oct. 2020), p. 16491. DOI: 10.1038/s41598-020-73339-y.

[31] Ayat Abedalla et al. *The 2ST-UNet for Pneumothorax Segmentation in Chest X-Rays using ResNet34 as a Backbone for U-Net*. en. arXiv:2009.02805 [cs, eess]. Sept. 2020. URL: http://arxiv.org/abs/2009.02805 (visited on 09/11/2024).

[32] Yuzhi Chen. “Application of Resnet18-Unet in separating tumors from brain MRI images”. en. In: *J. Phys.: Conf. Ser.* 2580.1 (Sept. 2023), p. 012057. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/2580/1/012057. URL: https://iopscience.iop.org/article/10.1088/1742-6596/2580/1/012057 (visited on 09/11/2024).

[33] Shruti Jadon. “A survey of loss functions for semantic segmentation”. en. In: *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. arXiv:2006.14822 [cs, eess]. Oct. 2020, pp. 1–7. DOI: 10.1109/CIBCB48159.2020.9277638. URL: http://arxiv.org/abs/2006.14822 (visited on 09/09/2024).

[34] Dominik Müller, Iñaki Soto-Rey, and Frank Kramer. “Towards a guideline for evaluation metrics in medical image segmentation”. en. In: *BMC Res Notes* 15.1 (June 2022), p. 210. ISSN: 1756-0500. DOI: 10.1186/s13104-022-06096-y. URL: https://doi.org/10.1186/s13104-022-06096-y (visited on 09/09/2024).

[35] Khaled Alomar, Halil Ibrahim Aysel, and Xiaohao Cai. "Data Augmentation in Classification and Segmentation: A Survey and New Strategies". en. In: *Journal of Imaging* 9.2 (Feb. 2023). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 46. ISSN: 2313-433X. DOI: 10.3390/jimaging9020046. URL: https://www.mdpi.com/2313-433X/9/2/46 (visited on 09/09/2024).

[36] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". en. In: *J Big Data* 6.1 (July 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0 (visited on 09/11/2024).

[37] Ekin D. Cubuk et al. *AutoAugment: Learning Augmentation Policies from Data*. en. arXiv:1805.09501 [cs, stat]. Apr. 2019. URL: http://arxiv.org/abs/1805.09501 (visited on 09/19/2024).

[38] Davood Karimi and Septimiu E. Salcudean. *Reducing the Hausdorff Distance in Medical Image Segmentation with Convolutional Neural Networks*. en. arXiv:1904.10030 [cs, eess, stat]. Apr. 2019. URL: http://arxiv.org/abs/1904.10030 (visited on 09/11/2024).

[39] Saeid Asgari Taghanaki et al. "Deep semantic segmentation of natural and medical images: a review". en. In: *Artif Intell Rev* 54.1 (Jan. 2021), pp. 137–178. ISSN: 1573-7462. DOI: 10.1007/s10462-020-09854-1. URL: https://doi.org/10.1007/s10462-020-09854-1 (visited on 09/11/2024).

[40] Nima Tajbakhsh et al. "Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation". In: *Medical Image Analysis* 63 (July 2020), p. 101693. ISSN: 1361-8415. DOI: 10.1016/j.media.2020.101693. URL: https://www.sciencedirect.com/science/article/pii/S136184152030058X (visited on 09/11/2024).

[41] Abolfazl Zargari, Najmeh Mashhadi, and S. Ali Shariati. "Enhanced Cell Tracking Using A GAN-based Super-Resolution Video-to-Video Time-Lapse Microscopy Generative Model". In: *bioRxiv* (June 2024), p. 2024.06.11.598572. ISSN: 2692-8205. DOI: 10.1101/2024.06.11.598572. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11195160/ (visited on 09/08/2024).

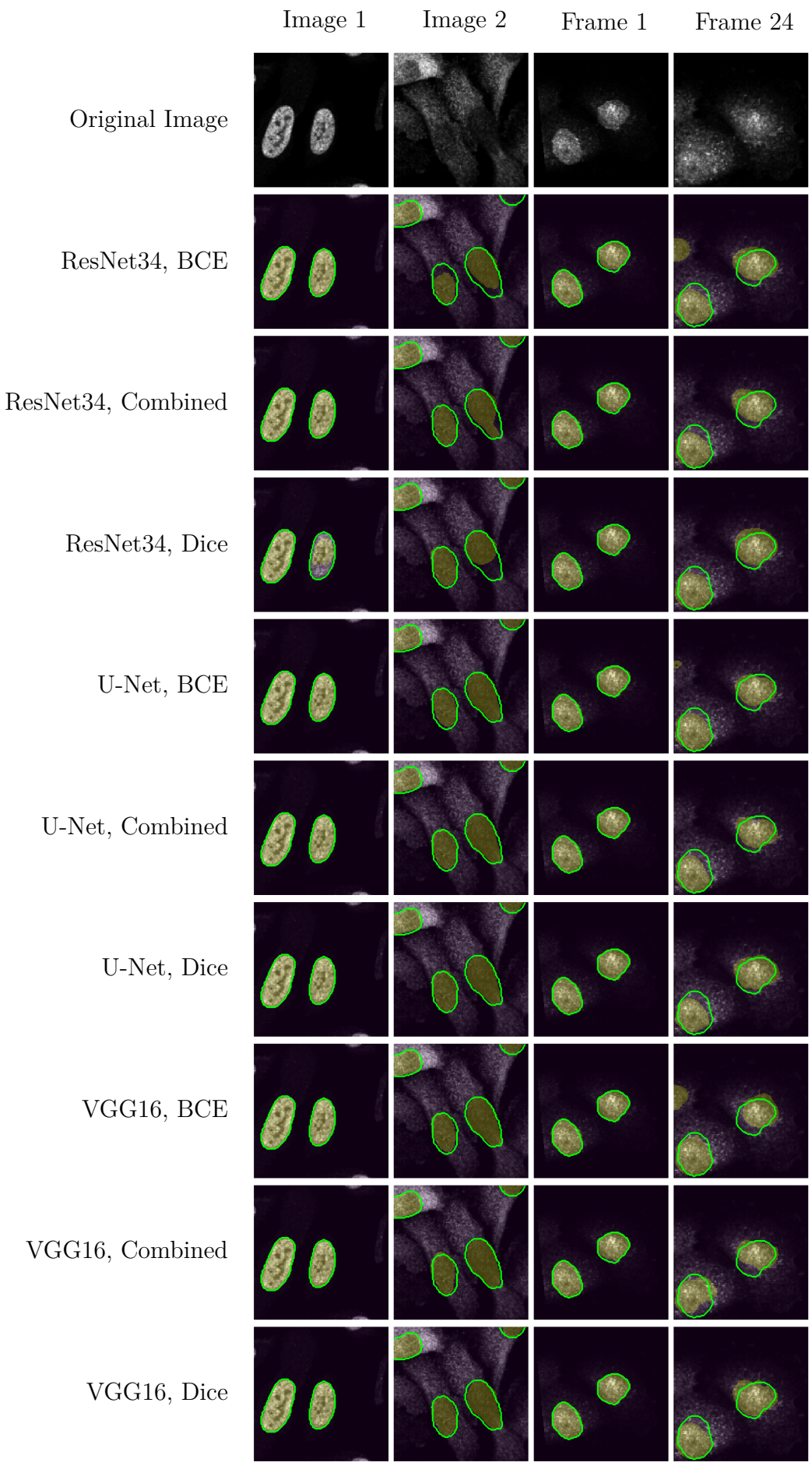
[42] Assaf Arbelle and Tammy Riklin Raviv. "Microscopy Cell Segmentation Via Convolutional LSTM Networks". In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. ISSN: 1945-8452. Apr. 2019, pp. 1008–1012. DOI: 10.1109/ISBI.2019.8759447. URL: https://ieeexplore.ieee.org/abstract/document/8759447?casa_token=bLGstoGSW5QAAAAA:5HUgG9b9WG4-9IIo2BorhFQz2KhCaTJX3Iasb32RDCoZ8KFBpYEX_1xqHbhlc (visited on 09/08/2024).

[43] Assaf Arbelle, Shaked Cohen, and Tammy Riklin Raviv. "Dual-Task ConvLSTM-UNet for Instance Segmentation of Weakly Annotated Microscopy Videos". In: *IEEE Transactions on Medical Imaging* 41.8 (Aug. 2022). Conference Name: IEEE Transactions on Medical Imaging, pp. 1948–1960. ISSN: 1558-254X. DOI: 10.1109/TMI.2022.3152927. URL: https://ieeexplore.ieee.org/abstract/document/9717246?casa_token=hP5ILVRx_WUAAAAA:cjr-BkZmHut6i3_mofTLsNY4_fYBh4M9HjpHhAbZwRXB (visited on 09/08/2024).

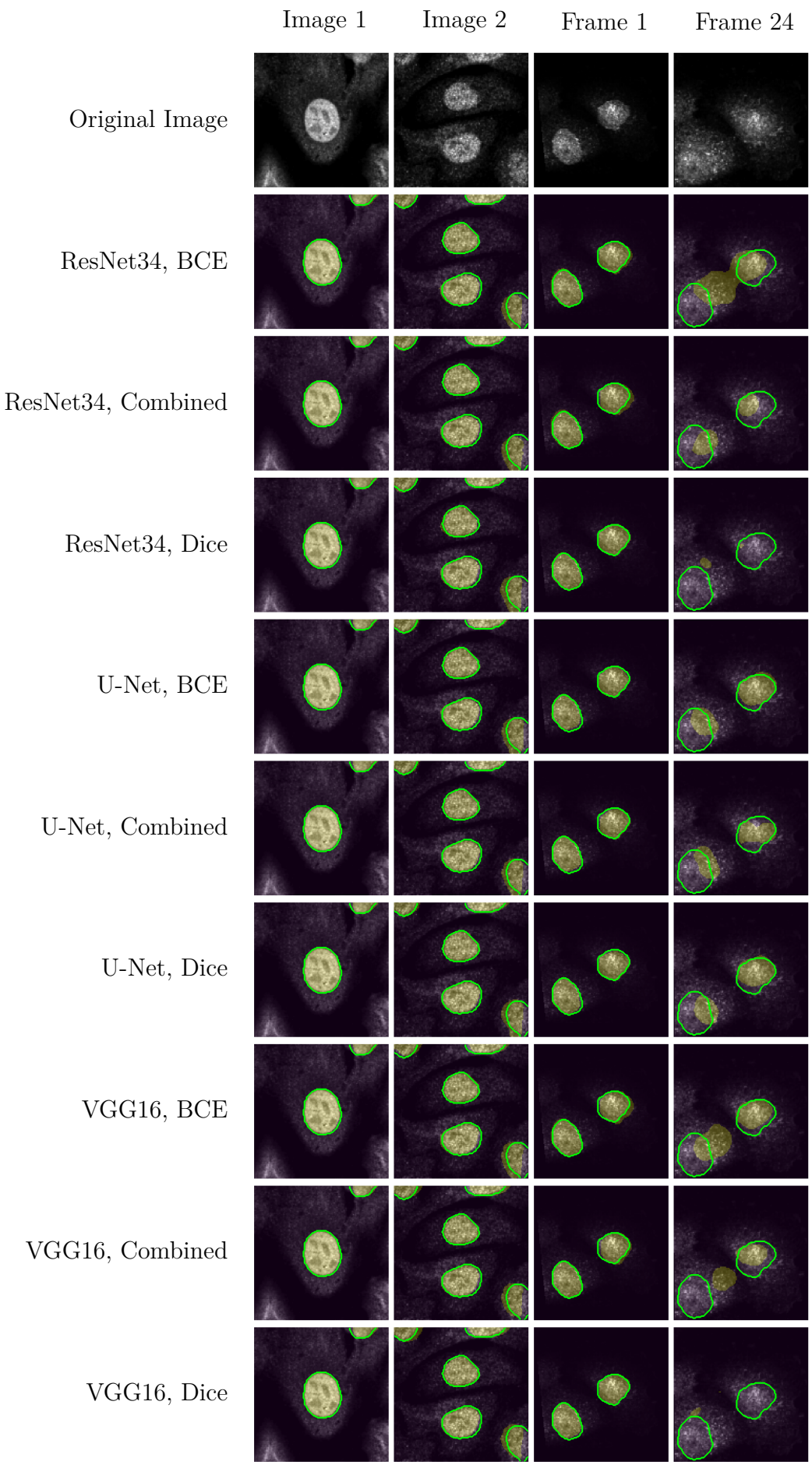
# 5 Supplementary Figures

The following figures show examples of model predictions against ground ground truth. The original images can be seen in the first row, each subsequent row showing the original image in purple, the corresponding model prediction in yellow, and the ground truth contours in green. Supplementary figures 1-3 show two random HPA validation examples, and the 1st and 24th frame of a synthetic HPA sequence. Supplementary figure 4 shows the 1st and 20th frame of two LEXY sequences.

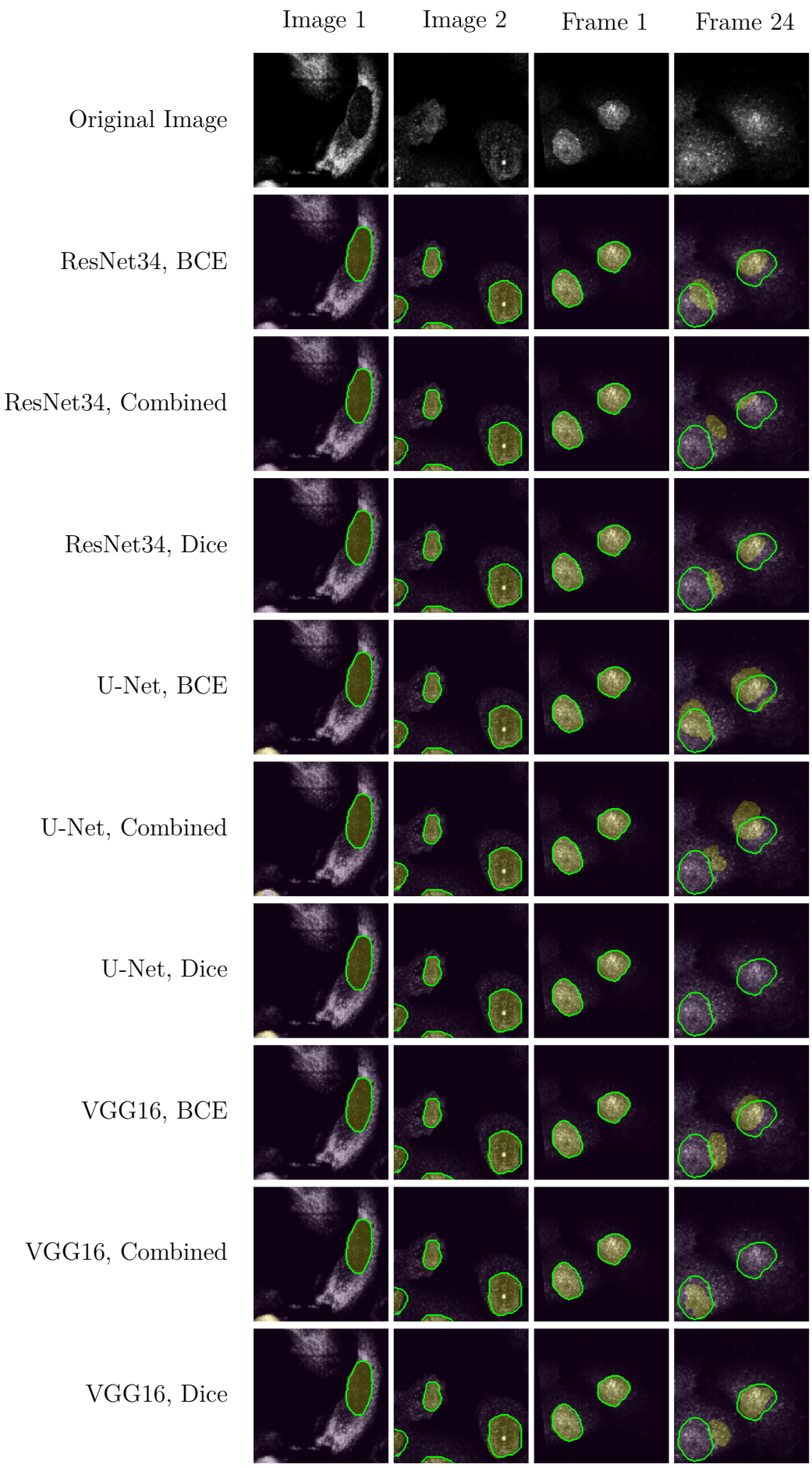
**Supplementary Figure 1: U-Net on HPA data**



**Supplementary Figure 2: U-Net using Dice loss on LEXY data**



**Supplementary Figure 3: Track-Net on HPA data**



**Supplementary Figure 4: U-Net using Dice loss on LEXY data**