Spatial-Temporal NAS for Fast Surgical Segmentation

Matthew Lee[1]
matt.lee3@medtronic.com
Felix Bragman[1]
felix.bragman@medtronic.com
Ricardo Sanchez-Matilla[1]
ricardo.sanchez-matilla@medtronic.com
Imanol Luengo[1]
imanol.luengo@medtronic.com
Danail Stoyanov[2]
ucl.ac.uk/surgical-robot-vision/people/danail-stoyanov

[1] Medtronic PLC,
230 City Road,
London, UK
EC1V 2QY

[2] University College London
Gower Street,
London, UK
WC1E 6BT

Abstract

Real time surgical video semantic segmentation requires accurate per-frame performance, inter-frame temporal consistency and high inference speeds. Additionally surgical guidance systems require a combination of spatial and temporal information to prevent distracting temporal effects such as flickering. Designing models that learn effective spatial-temporal representations poses two challenges. First, the relative importance of temporal and spatial features given an architecture and a dataset is unknown, requiring human intuition to design the model. Secondly, adding temporal information greatly increases model size, which can negatively affect its inference speed and may lead to overfitting. We propose ST-NAS, a novel Neural Architecture Search (NAS) framework for optimising the balance between spatial and temporal operations in spatial-temporal models. We introduce a regulariser that promotes faster inference speeds whilst balancing model performance. Components in the framework can be selectively used when considering the speed-accuracy requirements of the final model. We apply this framework to a private Partial Nephrectomy dataset and the public CholecSeg8K dataset. The model discovered through ST-NAS achieved a significant inference speedup (50-154%) with a marginal reduction in segmentation performance (1-5%). Experiments showed that the architecture discovered through ST-NAS required minimal temporal operations; supporting the effectiveness of architecture search in spatial-temporal network design.

1 Introduction

Automatic segmentation of surgical scenes can help train new surgeons [7], generate post-operative analytics, and intra-operatively highlight critical structures and surgical instruments [14]. Semantic segmentation of surgical videos is a key step to developing advanced digital tools for surgeons. For intra-operative use, they must work in real time and must be

temporally stable to prevent temporal artefacts from distracting surgeons. Many models have been applied to surgical segmentation [6, 8, 9]. However, when trained without appropriate temporal modelling, predictions can be unstable, resulting in poor temporal consistency and worse performance. Whilst the addition of temporal context can alleviate these problems, it also leads to a significant increase in computational cost and limits its ability to achieve fast inference speeds [6]. The model must be accurate enough to be useful for a surgeon, however this does not mean it requires perfect accuracy. Nor does it mean that improving accuracy is always worth the cost of losing other desirable attributes. If a model is too slow to process data in real time, it cannot be used intra-operatively, preventing surgeons from gaining any benefit. A slow model will also risk lagging behind the data stream, causing the model to be unusable in real-time settings. For these reasons, we believe sacrificing accuracy for inference speed in a controlled environment can sometimes be valuable even in the surgical and medical imaging domains.

Designing spatial-temporal architectures generally relies on human intuition to determine the balance between spatial and temporal features. Naïvely using convolutions with temporal context can allow the architecture to learn spatial-temporal representations, but is computationally expensive and often leads to overfitting [10, 18]. The relative importance of spatial vs temporal information is unknown without careful analysis [10]. Without this knowledge, it is not possible to design optimal spatial-temporal architectures whilst a grid-search over possible connectivity patterns is often computationally intractable.

Neural Architecture Search (NAS) has emerged as a solution to automate the search for optimal architectures over large search spaces in computer vision [4]. Differentiable Neural Architecture Search (DNAS) presents a computationally feasible approach, e.g. FBNet [19]. This motivates our use of DNAS to learn spatial-temporal architectures for video semantic segmentation. In this work, we propose Spatial Temporal NAS (ST-NAS), a differentiable NAS framework for spatial-temporal models, which balances both accuracy and inference time when seeking the optimal architecture. ST-NAS can be applied to transformer backbones through an inference time regulariser. Our framework further introduces the use of DNAS for optimising spatial-temporal convolutions in a temporal decoder. ST-NAS search is lightweight and requires less than 10% of the total training time, making it a suitable framework for training video segmentation models.

2 Related Work

Surgical video segmentation is an important surgical data analysis task and temporal modelling ensures robustness against occlusions. [6, 9] developed models to improve temporal consistency. A spatial-temporal decoder was presented using a Temporal Convolutional Network (TCN) [6], whilst a novel hierarchical transformer employing a space-time shift was presented in [9]. However, both methods depend largely on manual design decisions such as in the TCN decoder of [6]. Our method enables us to learn the design of the TCN and the balance between spatial and temporal representations (Section 3.3).

Various general DNAS methods have been developed to enable efficient architecture search [11, 19] with particular success when applied to dense vision prediction tasks using CNN backbones [19, 23]. However, NAS has seen less success when applied to transformer architectures, which are competing with CNNs as computer vision backbones. For instance, Swin transformer [12] produces hierarchical representations and relies on shifted-windows that have complexity linear with image size. However, we show that it suffers from poor

latency at inference (Table 1), so Swin is a prime candidate to be optimised through NAS. Evolutionary NAS methods [1] have been applied to transformers but are computationally expensive. Autoformer [1] searches for an optimal transformer through weight sharing and weight entanglement to reduce the search space but used an evolutionary algorithm with high computational cost. NASBert [20] required hundreds of GPU days for initialisation. This makes the use of DNAS and the block based search of FBNet [19] attractive for developing NAS methods and motivates the work presented in this manuscript (Section 3.1).

In the context of semantic segmentation from videos, attention has focused largely on static models [2, 13] and there is a paucity of NAS frameworks for tackling spatial-temporal models for semantic segmentation, especially those which depend on the use of TCNs in decoders. TCNs have been used extensively in temporal models [5, 6] but suffer generally from computational overheads and overfitting [10, 18] when applied to static encodings. There have been NAS approaches towards temporal models in action recognition [16] and signal processing [17]. NAS-TC [16] searches for optimal temporal convolutions but its search space is limited to changes in spatial feature extraction and temporal operations are handed-crafted; limiting its ability to generalise. In comparison, our learning framework automatically searches for optimal temporal representations (Section 3.3) through the depth of the encoder but also the necessary temporal operations.

3 Methods

We present Spatial-Temporal NAS (ST-NAS), a novel learning scheme to search for the optimal spatial-temporal network for video semantic segmentation. Our aim is to automatically search for an architecture with reduced inference time and able to maintain high accuracy relative to a baseline architecture. Our framework is applied to spatio-temporal models with an encoder-decoder architecture such as in [6] which uses a Swin Encoder to generate per-frame features for a temporal TCN decoder. The encoder $E(\cdot)$ processes a temporal sequence of RGB frames $\mathbf{x}_t \in \mathbb{R}^{3\times H\times W}$ where t indexes the time in a sequence $T \in \{t-w,t\}$ with a temporal window w. The encoder processes each frame t sequentially such that $\mathbf{f}_t = E(\mathbf{x}_t)$, where $\mathbf{f}_t \in \mathbb{R}^{L\times H\times W}$ is a spatial feature representation and L is the number of feature maps. A temporal decoder $D(\cdot)$ with TCNs processes the temporal feature batch $\mathbf{f}_T$ and predicts the segmentation mask $\mathbf{s}_t = D(\mathbf{f}_T)$ for time t.

Our NAS learning scheme simultaneously tackles the inference speed of transformer-based encoders whilst discovering the optimal balance of temporal operations in a decoder. In Section 3.2, we use NAS to discover the optimal depth of the encoder to improve inference speed whilst balancing the quality of its feature representations. In Section 3.3 we develop the NAS framework for learning the optimal combination of spatial and temporal operations to achieve enhanced segmentation performance.

3.1 Differentiable NAS

The search space of a NAS algorithm defines a set of candidate architectures discoverable by the algorithm. The search space in DNAS is formed by creating one architecture that represents the entire search space, which is defined as a *supernet*. We adopt a block based layerwise search space similar to FBNet [19], in which the supernet is formed of searchable blocks but unlike FBNet, all candidate operations are run in parallel rather than single operations being sampled at each step, as in DARTS [11]. The output of each candidate operation

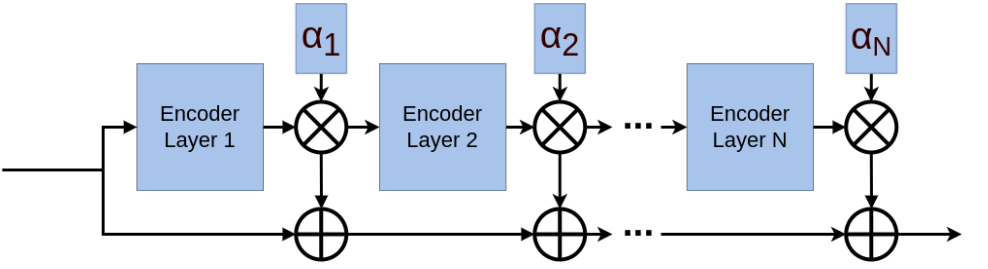

Figure 1: We convert the encoder to a supernet. Each layer in the stage is weighted and the output becomes the weighted sum of each layers output.

is weighted by its own architectural weight, α_i^j where i indexes the supernet layer and j the operation. This α_i^j weights each operation within the supernet. Whilst the supernet is trained, all network parameters are optimised, including all operations within each supernet block and the set of all $\mathbf{A} = \{\alpha_i^j\}_{\forall i}^{\forall j}$ values. During training, the supernet learns to increase the α_i^j for optimal operations that minimise the loss function, and learns to decrease the α_i^j for sub-optimal operations. At the end of training, the final proposed architecture is chosen by selecting only the operations with the largest $\alpha_i = \arg\max_j \left(\alpha_i^j\right)$ for each layer i. DNAS consequently searches an combinatorial architecture space in tractable time.

3.2 Encoder NAS for speed

To improve the inference speeds while achieving competitive accuracy, we propose to apply DNAS to the encoder. Typically, DNAS is applied to select the optimal operations, network depth and network width; however, as our goal is to reduce inference time and keep search cost low, we limit our search space to find only the optimal depth. Limiting the search space can reduce search cost and improve final performance [15]. To find the optimal depth of an encoder composed of sequential blocks, we modify the traditional DNAS to form a supernet by taking the original encoder architecture and converting each block within it as depicted in Fig 1. For each i-th block, a weight α_i is assigned with the aim of learning the importance of the block. Then, the output of the block is the weighted sum of the layer outputs instead of the output of the final layer. This is similar to the concept of applying skip connections, but with the addition of the architectural weights. This novel concept can potentially be applied to any encoder with a sequential structure of blocks. To formalise this concept, a traditional encoder with sequential blocks can be written as $\mathbf{f} = E(\mathbf{x}) = e_i(e_{i-1}(\ldots e_2(e_1(\mathbf{x}))\ldots))$, where N is the number of blocks of the encoder. We propose to convert the previous to

$$\mathbf{f} = E(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i f_i = \sum_{i=1}^{N} \alpha_i e_i(\alpha_{i-1} e_{i-1}(\ldots \alpha_2 e_2(\alpha_1 e_1(\mathbf{x}))\ldots) \tag{1}$$

where f_i is the feature map produced by layer i-th block, and α_i is its architectural weight. Note that the temporal component t has been omitted for clarity in the above equation.

Each layer within a stage uses the outputs of the previous layer, meaning we can only remove layers from the end of the stage. During the depth shrinking process, we consider each a layer, starting from the last (N) and move towards the first (N-1, N-2, ..., 1). We

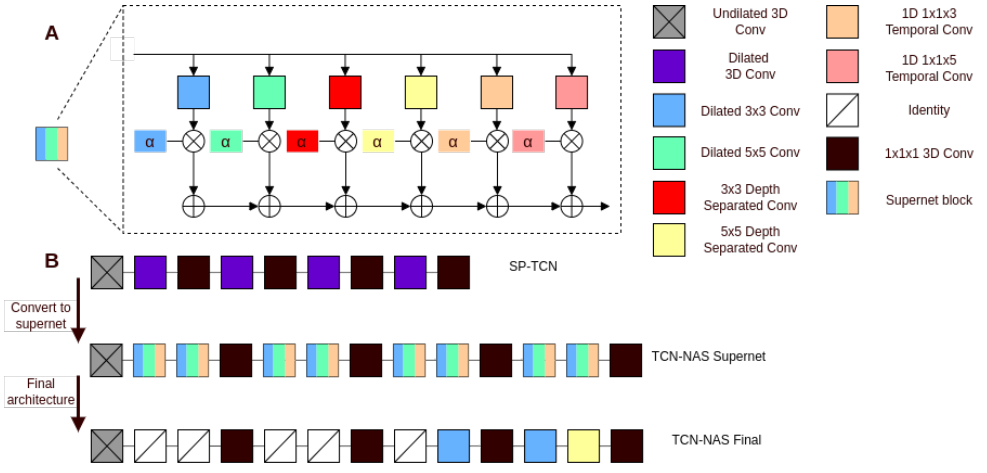

Figure 2: ST-NAS searches for the optimal spatial and temporal operations of each layer of the TCN. a) To convert to a NAS architecture, the 3D convolutions are replaced with with supernet blocks. Within each supernet block candidate operations are applied in parallel their outputs are combined via weighted sum. We replace each 3D conv with two supernet blocks, to allow equivalent spatial-temporal operations. The weighting α is learnt during training. b) Once the training is complete, the operation with the largest α is kept and the others are discarded. This results in the final architecture.

remove the i-th layer if $\alpha_i < 1$ and continue to the $i-1$th layer. If the layer's $\alpha_i \geq 1$, we stop the algorithm and the remaining layers form the final encoder architecture. We apply this algorithm to each stage of the encoder.

When training the supernet encoder in this way, the α_i values will naturally increase with depth (i) as subsequent layers will further refine the features and provide more utility. To account for the cost of these layers during the supernet learning, we add an L1 regularisation term to the loss. For each layer, the inference time for that layer is calculated. We regularise the alpha weights as the weighted sum of layer inference times. In practise, this encourages the discovery of architectures by regularising proportionally to their inference time cost. Such regularisation has been effective in other DNAS methods [17, 19].

$$\min_{\mathcal{W},\mathbf{A}} \mathcal{L}(f(\mathcal{W},\mathbf{A},\mathbf{x}),\mathbf{y}) + \lambda \sum_{i=1}^{N} \alpha_i s_i, \tag{2}$$

where $\mathcal{L}(\cdot)$ is the loss function, f is the model's forward pass operation, $\mathcal{W}$ are the weights of the model, $\mathcal{A}$ is the architecture of the model, x are the input data, y are the ground truth segmentation masks, λ is a regularisation weight, and s_i is the inference time for that block. Note that while larger values of λ will produce faster architectures but with lower accuracy, smaller values will produce architectures with higher accuracy but slower inference. By weighting with the α, we allow the regularisation to influence the architecture during DNAS.

3.3 Decoder NAS for performance

The decoder in a spatial-temporal model combines spatial and temporal information, often using 3D convolutions. Decomposing 3D convolutions into their spatial and temporal convolutions can improve performance and greatly reduce computational cost [13]. In addition, the importance of spatial and temporal information varies between datasets. We propose to tackle this problem by modifying the decoder and searching for the optimal sequence of spatial and temporal convolutions to apply. Specifically, we apply a simple differentiable NAS approach. We define a set of candidate operations, including spatial and temporal operations: dilated 2D spatial conv, depth separated 2D spatial conv, dilated 1D temporal conv and the identity function. We use kernel sizes of 3 and 5, bringing the total operations to $3 \times 2 + 1 = 7$. At each layer i, we apply all the candidate operations in parallel and output the weighted of sum of their outputs. This forms a supernet, Fig 2, as described in Section 3.1. Similar to Section 3.2, the weighting of each output is based on that operation's α_i^j, however the encoder and decoder NAS steps are performed separately. We train the supernet and then trim all operations except for those with the highest α weights at each layer. This results in a novel decoder that uses 2D spatial and 1D temporal convolutions.

4 Experiments and Results

Datasets. We validate our approach with two datasets; a private dataset of image sequences from partial nephrectomy (PN) procedures and the publicly available CholecSeg8K dataset which contains image sequences from cholecystectomy procedures [8]. The PN dataset consists of 53,000 images from 137 procedures annotated with segmentation masks for four classes (kidney, liver, renal vein, and renal artery). Short video clips of 15 seconds were annotated at 10 FPS. The images were labelled by trained non-medical experts under the supervision of an anatomy specialist, using annotation guidelines validated by surgeons.The CholecSeg8K dataset consists of 8080 images from 17 videos of the Cholec80 dataset annotated at 25FPS[8]. Images are annotated with segmentation masks containing 13 classes (background, abdominal wall, liver, gastrointestinal tract, fat, grasper, connective tissue, blood, cystic duct, l-hook electrocautery, gallbladder, hepatic vein and liver ligament). For both datasets we used the same train/val splits defined in [6].

Metrics. We assess the model's inference speed with Frames Per Second (FPS) and the segmentation accuracy with the Mean Intersection Over Union (mIOU). FPS was calculated from inference time. $FPS = 1/T_{inf}$ Inference time was measured as time to encode one image plus time to decode a full 11 frame encoding window. i.e. the time to process one frame in a real application. The inference time was averaged over 1000 frames. The floating point operations (FLOP) count is commonly used to measure a model's computational cost. However, during inference, factors such as parallelisation and GPU synchronisation can also impact the speed. To avoid confusion, we do not report FLOP, as it does not directly correlate with inference speed.

Model. The NAS framework is applied to the spatial-temporal model of [6]. The original model used Swin-B as the encoder which we provide as a baseline. However, as our objective is to accelerate the model, we extend from the Swin-T backbone. Swin-T has fewer layers and a smaller embedding dimension than Swin-B. The Swin transformer is comprised of 4 stages. Each stage is converted to make the decoder a supernet as described in Section 3.2. We train this supernet encoder on the PN dataset for just 5 epochs to learn these α values

Table 1: Results on PN and CholecSeg8K datasets. Our proposed model uses TCNNAS decoder, Full Channels (FC) and our shortened Swin encoder to significantly increase FPS. Full Channels (FC) removes the channel reduction layer between the encoder and decoders.

Model	Swin Encoder	TCNNAS	FC	Params	FPS	PN mIOU	CholecSeg8K mIOU
Swin-B †	Base	×	×	95.9M	14.7	60.3	53.2
Swin-T †	Tiny	×	×	34.4M	23.1	60.5	52.3
NAS-Unet [23] †	-	×	×	13.9M	34.8	27.1	40.4
HR-NAS [3] †	-	×	×	10.2M	9.4	36.3	42.2
SP-TCN [6]	Base	×	×	102.6M	12.6	70.7	57.0
	Tiny	×	×	40.4M	20.5	73.9	54.3
Swin Short	Short	×	×	28.9M	24.8	72.3	51.0
Full Channels	Short	×	✓	33.0M	28.0	68.3	50.8
ST-NAS (Ours)	Short	✓	✓	39.0M	**31.9**	69.5	51.5

†Non-temporal model. i.e. time window=1

and then find the final model, which we name Swin Short. Swin Short stages have depths of 1,2,3,2 shortened from the Swin-T's 2,2,6,2. The final model is then trained for 25 epochs.

We additionally remove the expensive 3D convolution applied between the encoder and decoder used to reduce the channels. We refer to this as the full channel (FC) modification. Allowing full channels to enter the SP-TCN costs little but saves time on the expensive channel reducing convolution. For the decoder, SP-TCN uses dilated 3D convolutions. We convert the SP-TCN to a supernet as described in Section 3.3. We train this supernet decoder on the PN dataset for 5 epochs to find the final model which we name TCNNAS. The discovered architecture is shown in Fig 2. It uses no temporal convolutions and so the only temporal processing is performed by the initial 3D conv that was not part of the dilated TCN layers and therefore not part of the search. The architectures discovered for the encoder and decoder on PN were then applied to CholecSeg8K. This was required as CholecSeg8K is much smaller and we found that even the baseline models tended to overfit. ST-NAS introduced more parameters, causing more overfitting and the NAS performed poorly.

Experimental setup. Models were trained on V100 GPUs using PyTorch 2.0.1. The models were trained with: Batch size 16, learning rate 10^{-4}, time window of 11, cross entropy loss, AdamW optimiser with 10^{-3} weight decay and 0.9 momentum. Images were resized to width 576, height 320 pixels with 3 colour channels. Images were augmented with random {-10,10} degree rotation and horizontal flips .

Results. Our results are shown in Table 1 and Fig 3. We provide baselines to compare to [6] as newer versions of PyTorch have altered performance. Fig 3. shows the effect of the encoder NAS on the inference speed (brown arrow). A considerable 21% FPS gain is made with a small drop in performance (-1.6% PN, -3% CholecSeg8K). To further develop the FPS, we analysed the inference time of layers within the model and found an expensive channel reduction between the encoder and decoder. The effect of removing and using full channels is shown in Fig 3 as the red arrow. This manual change is specific to the SP-TCN base model, but it gives a 13% increase in FPS, although with a mIOU reduction (-4% PN, -0.2% CholecSeg8K). TCNNAS is shown as the grey arrow in Fig 3. TCNNAS recovers some lost mIOU (+1.2% PN, +0.7% CholecSeg8K) and further boosts the model FPS (+14%).

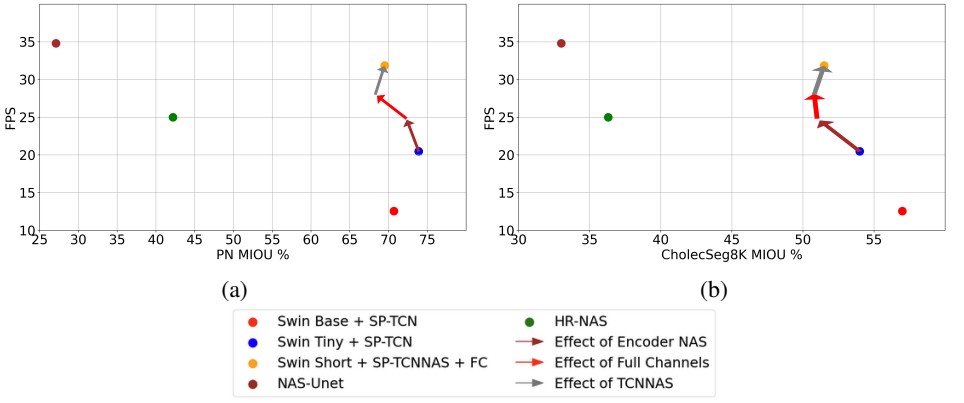

Figure 3: (a) mIOU vs FPS for PN dataset. (b) mIOU vs FPS for CholecSeg8K dataset. Arrows indicate the effect of each adaptation step. Effect of Encoder NAS: Change from applying NAS to the encoder of the model. Effect of Full Channels: Change from passing all channels to decoder. Effect of TCNNAS: Change from applying NAS to our TCN-based decoder. NAS-Unet and Swin Base + SP-TCN are shown as comparison baselines. NAS-Unet has a slightly faster FPS, but with a greatly reduced mIOU.

For PN, an initial improvement to speed and mIOU is given by simply using Swin-T as the encoder, therefore we compare our models to Swin-T. Together, our modifications increased speed 56% and only reduced mIOU 4.4% in comparison to Swin-T + SP-TCN.Regarding CholecSeg8K, Swin-T baseline performed worse than the Swin-B, therefore we compare with Swin-B + SP-TCN. Our final method provides a 2.54× speedup but with a significant drop in mIOU (4.5%) in comparison to Swin-B. Compared to Swin-T + SP-TCN we see a 56% speedup with a 1.5% reduction in mIOU. A better compromise for CholecSeg8K, might be to use FC and TCNNAS with Swin-T encoder, which gets the best performance on CholecSeg8K but is 67% faster than Swin-B + SP-TCN.

NAS-Unet had the highest FPS but reached relatively low MiOU of 27.1% and 40.4% on PN and CholecSeg8k respectively, possibly because it's smaller size and U-Net backbone lacked the expressivity required for this task. HR-NAS was similarly unable to achieve high MiOU scores but also suffered from the lowest FPS. Both NAS-Unet and HR-NAS models were non-temporal but were unable to match the MiOU of the Swin non-temporal model baselines.

We benchmarked our model against NAS methods [3, 22] and strong baselines such as Swin [12] and SP-TCN [6]. We have demonstrated consistently superior performance compared to NAS-Unet and HR-NAS whilst discovering an architecture more amenable to real-time deployment. NAS-UNet [22] achieves high FPS whilst significantly underperforming compared to our method. Similarly, HR-NAS [3] performed poorly compared to our method and yields an architecture unsuitable for real-time use. When compared to Swin-Tiny [12] and SP-TCN-Tiny [6], our method made a significant improvement in throughput with marginal loss in performance; resulting in an architecture optimised for real-time use. The slight decrease in MiOU has a minimal effect on the qualitative changes (Fig. 4). However the improved FPS of our model is clear in our supplementary video. We found that using a non-causal time window, i.e. with future frames: $T \in \{t-w/2, t+w/2\}$ had no significant effect on results. Training our final model with a causal TCN increased performance by

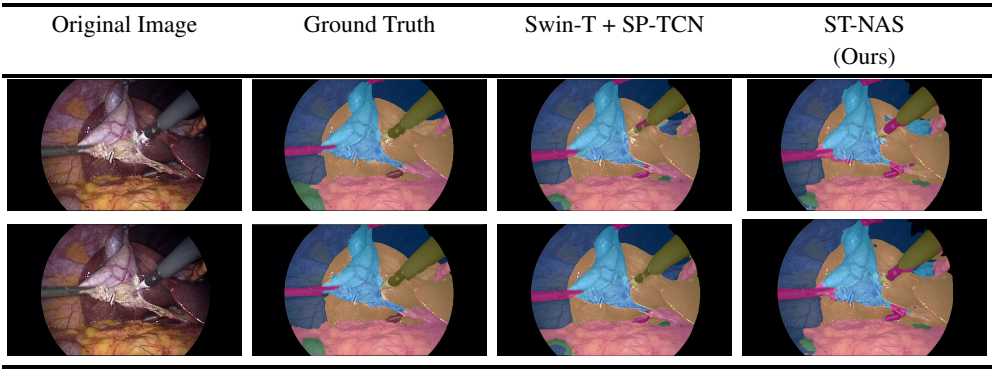

Figure 4: Comparisons between our method and Swin-T SP-TCN for semantic segmentation of CholecSeg8K images [8]. Second row is the next frame in the sequence. As we are aiming to accelerate the network while maintaining performance, there is little difference between the qualitative appearance of the outputs. Background, Abdomen wall, Grasper, Electrocautery, Liver, Fat, Gallbladder, Gastrointestinal tract and Hepatic vein.

0.7% MiOU on CholecSeg8K, illustrating suitability for causal or non-causal applications.

Training time. Our NAS method required less than an hour to search for the correct encoder depth and 1.8hrs to search for the optimal decoder operations. These costs were offset by the faster training time of Swin Short and TCNNAS. Swin Short trained in 23.5hrs and Swin-T in 26.1hrs. The TCNNAS and SP-TCN required 5hr and 6.9hrs respectively. This means our overall method was trained 2hrs (6%) faster than Swin-T + SP-TCN and the NAS search only cost 9.7% of our overall training time. The other NAS benchmarks NAS-Unet and HR-NAS required 9.7hrs and 7.4hrs respectively.

5 Discussion

ST-NAS is 56% faster by optimising encoder depth and the balance of spatial to temporal operations in the decoder, with marginal reductions in mIOU of 4.6% on PN. ST-NAS finds the right balance between temporal and spatial information for a spatio-temporal model. In fact, the experiments revealed that using minimal temporal operations obtains better results, which is counter-intuitive, showing the value of NAS over architecture design lead by human intuition.

The decoder architecture discovered is shown in Fig 2. This architecture used several identity functions in its early layers, followed by depth separated and dilated spatial convolutions. Interestingly, no temporal convolutions were used by the TCNNAS. This means that other than the initial 3x3x3 conv used as the first layer of the TCNNAS, there are no other temporal elements. This shows only lightweight temporal operations are required for these tasks, saving the use of computationally expensive operations seen in the baseline. However, temporal information is important as non-temporal versions of these models (i.e. Swin-T) performed worse as seen in table 1. This suggests designing spatial-temporal models by human intuition might not be optimal, as the importance of temporal information may be counter-intuitive [10].

ST-NAS provides a lightweight and efficient training scheme, decreasing total training

time in comparison to the baseline temporal models. In contrast, existing methods such as AutoFormer [1] required extensive training of 500 epochs for their respective supernet training. This makes ST-NAS a suitable option for low-resources settings that cannot afford expensive supernet training. We found that models discovered by architecture search with PN data, can be applied effectively to CholecSeg8K. We also found that CholecSeg8K annotations were inconsistent over time, with some annotations varying greatly from frame to frame. We believe this added to the difficulty in training a temporal model on CholecSeg8K. DNAS methods can find non-optimal architectures, with lower-parameter operations often being favoured due to them converging faster [21]. To mitigate these issues, we used warm-up epochs and inference time regularisation. Ultimately this was less of a problem for our use case, where smaller operations were desired for faster inference.

Conclusion In this paper, we propose ST-NAS, a DNAS algorithm for improving FPS on spatial-temporal models. Our final models were 56% faster than the baseline with a small 4.6% drop in mIOU. Our NAS search was cheap and in fact reduced overall training time for the models. Future work should search for the optimal width of the encoder layers to find further optimisations.

References

[1] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. pages 12250–12260, 2022.

[2] Wuyang Chen et al. Fasterseg: Searching for faster real-time semantic segmentation. In *International Conference on Learning Representations*, 2019.

[3] Mingyu Ding et al. Hr-nas: Searching efficient high-resolution neural architectures with lightweight transformers. pages 2981–2991, 2021. ISBN 9781665445092.

[4] Thomas Elsken et al. Neural architecture search for dense prediction tasks in computer vision, 2022.

[5] Yazan Abu Farha and Jürgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *CVPR*, pages 3570–3579, 2019.

[6] Maria Grammatikopoulou et al. A spatio-temporal network for video semantic segmentation in surgical videos. *IJCARS*, pages 1–8, 6 2023. ISSN 18616429.

[7] David T. Guerrero, Malke Asaad, Aashish Rajesh, Abbas Hassan, and Charles E. Butler. Advancing surgical education: The use of artificial intelligence in surgical training. *American Surgeon*, 89:49–54, 1 2023. ISSN 15559823.

[8] W. Y. Hong et al. Cholecseg8k: A semantic segmentation dataset for laparoscopic cholecystectomy based on cholec80. 2020.

[9] Yueming Jin et al. Exploring intra- and inter-video relation for surgical semantic scene segmentation. *IEEE Transactions on Medical Imaging*, 41(11):2991–3002, 2022.

[10] Matthew Kowal et al. A deeper dive into what deep spatiotemporal networks encode: Quantifying static vs. dynamic information, 2022.

[11] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv*, 6 2018. ISSN 23318422.

[12] Ze Liu et al. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[13] Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Architecture search of dynamic cells for semantic video segmentation. In *WACV 2020*, pages 1959–1968, 2020.

[14] David Owen, Maria Grammatikopoulou, Imanol Luengo, and Danail Stoyanov. Automated identification of critical structures in laparoscopic cholecystectomy. *IJCAR*, 17: 2173–2181, 12 2022. ISSN 18616429. URL https://link.springer.com/article/10.1007/s11548-022-02771-4.

[15] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. pages 10425–10433, 2020.

[16] Pengzhen Ren et al. Nas-tc: Neural architecture search on temporal convolutions for complex action recognition. 3 2021.

[17] Matteo Risso et al. Lightweight neural architecture search for temporal convolutional networks at the edge. *IEEE Transactions on Computers*, 72:744–758, 1 2023.

[18] Du Tran et al. A closer look at spatiotemporal convolutions for action recognition. *CVPR*, pages 6450–6459, 11 2017. ISSN 10636919.

[19] Bichen Wu et al. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *CVPR*, 2019-June:10726–10734, 2019. ISSN 10636919.

[20] Jin Xu et al. Nas-bert: Task-agnostic and adaptive-size bert compression with neural architecture search. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 11:1933–1943, 5 2021.

[21] Arber Zela et al. Understanding and robustifying differentiable architecture search. 2019.

[22] Tianbao Zhou, Y U Weng, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. 2019.

[23] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CVPR*, pages 8697–8710, 2018. ISSN 10636919.