Evolving Space-Time Neural Architectures for Videos

AJ Piergiovanni, Anelia Angelova, Alexander Toshev, Michael S. Ryoo
Google Brain
{ajpiergi,anelia,toshev,mryoo}@google.com

Abstract

We present a new method for finding video CNN architectures that capture rich spatio-temporal *information in videos. Previous work, taking advantage of 3D convolutions, obtained promising results by manually designing video CNN architectures. We here develop a novel evolutionary search algorithm that automatically explores models with different types and combinations of layers to jointly learn interactions between spatial and temporal aspects of video representations. We demonstrate the generality of this algorithm by applying it to two meta-architectures, obtaining new architectures superior to manually designed architectures: EvaNet. Further, we propose a new component, the iTGM layer, which more efficiently utilizes its parameters to allow learning of space-time interactions over longer time horizons. The iTGM layer is often preferred by the evolutionary algorithm and allows building cost-efficient networks. The proposed approach discovers new and diverse video architectures that were previously unknown. More importantly they are both more accurate and faster than prior models, and outperform the state-of-the-art results on multiple datasets we test, including HMDB, Kinetics, and Moments in Time. We will open source the code and models, to encourage future model development* [1].

1. Introduction

Video understanding tasks, such as video object detection and activity recognition, are important for many societal applications of computer vision including robot perception, smart cities, medical analysis, and more. Convolutional neural networks (CNNs) have been popular for video understanding, with many successful prior approaches, including C3D [30], I3D [1], R(2+1)D [33], S3D [38], and others [3, 9]. These approaches focus on manually designing CNN architectures specialized for videos, for example by extending known 2D architectures such as Inception [28] and ResNet [5] to 3D [1, 33]. However, designing new, larger or more advanced architectures is a challenging problem, especially as the complexity of video tasks necessitates deeper and wider architectures and more complex sub-modules. Furthermore, the existing networks, which are mostly inspired by single-image based tasks, might not sufficiently capture the rich spatio-temporal interactions in video data.

In this work, we present a video architecture evolution approach to harness the rich *spatio-temporal* information present in videos. Neural architecture search and evolution have been previously applied for text and image classification [29, 41]. A naive extension of the above approaches to video is infeasible due to the large search space of possible architectures operating on 3D inputs.

To address these challenges we propose a novel evolution algorithm for video architecture search. We introduce a hybrid meta-architecture ('fill-in-the-blanks') model for which the high level connectivity between modules is fixed, but the individual modules can evolve. We apply this successfully to both Inception and ResNet based meta-architectures. We design the search space specifically for video CNN architectures that jointly capture various spatial and temporal interactions in videos. We encourage exploration of more *diverse* architectures by applying multiple nontrivial mutations at the earlier stages of evolution while constraining the mutations at the later stages. This enables discovering multiple, very different but similarly good architectures, allowing us to form a better ensemble by combining them.

Furthermore, to enrich the search space for video inputs, we propose a new key element which is specifically designed to capture space-time features' interactions. We introduce an *Inflated Temporal Gaussian Mixture* (iTGM) layer as part of the evolution search space. The iTGM is motivated by the original 1D TGM [21]. For our iTGM, we learn 2D spatial filters in addition to the temporal Gaussian mixture values, and inflate the 2D filter temporally to allow learning of joint features in 3D. The 2D filter is inflated non-uniformly, by following the weights according to the learned 1-D temporal Gaussian mixture pattern. This allows to explore space-time interactions more effectively and with much fewer parameters, while at the same time capture longer temporal information in videos.

[1] Code and models: https://sites.google.com/corp/view/evanet-video

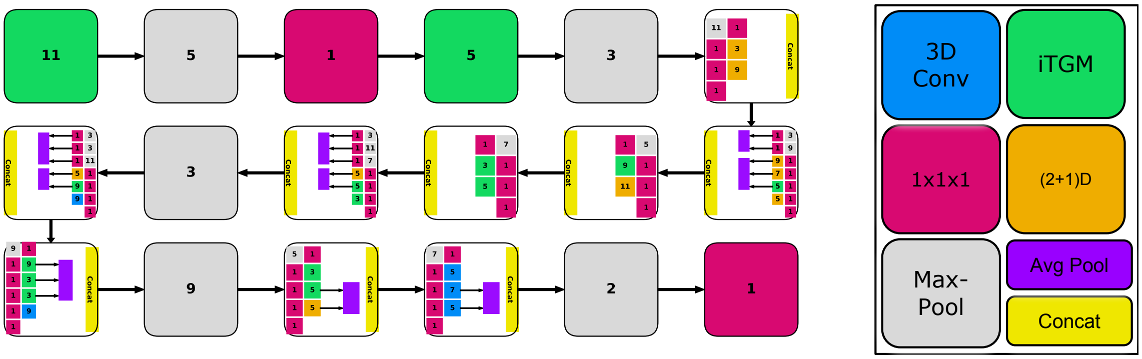

Figure 1. Example of a video architecture obtained with evolution. Inception-like architecture. The color encodes the type of the layer, as indicated on the right. The numbers indicate the temporal size of the filters in each module. See text for discussion.

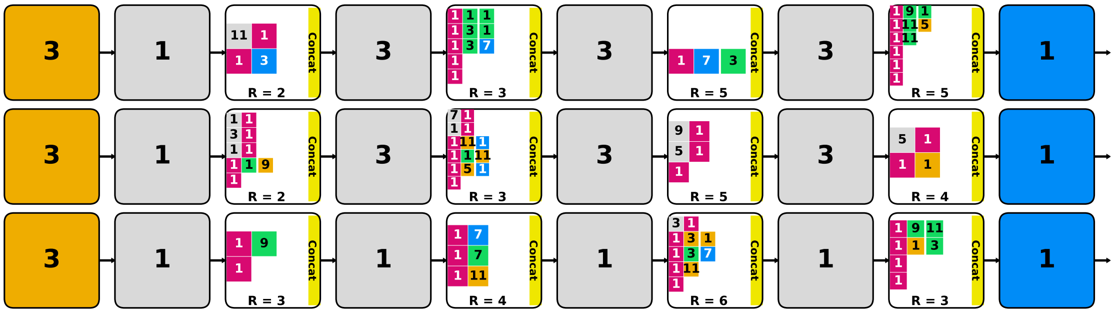

Figure 2. Three different ResNet-like architectures obtained for the Kinetics dataset. Modules are repeated R times.

The proposed algorithm results in novel architectures which comprise interesting sub-modules (see Fig. 1 and 2). It discovers complex substructures, including modules with multiple parallel space-time conv/pooling layers focusing on different temporal resolutions of video representations. Other findings include: multiple different types of layers combined in the same module e.g., an iTGM layer jointly with (2+1)D convolutions and pooling layers; heterogeneous modules at different levels of the architecture, which is in contrast to previous handcrafted models. Furthermore, the evolution itself generates a diverse set of accurate models. By ensembling them, recognition accuracy increases beyond other homogeneous-architecture ensembles.

Our approach discovers models which outperform the state-of-the-art on all four public datasets we tested (i.e., HMDB, Charades, Moments in time and Kinetics). This is done with a generic evolutionary algorithm and no per-data hyperparamter tuning. Furthermore, the best found models are very fast, running at about 100 ms for a single model, and 250ms for an ensemble, both being considerably faster than prior models.

The main technical contributions of this paper are: 1) We propose a novel evolutionary approach for developing space-time CNN architectures, specifically designed for videos. We design the search space to specifically explore different space-time convolutional layers and their combinations and encourage diversity. 2) We introduce a new space-time convolutional layer, the Inflated TGM layer, designed to capture longer-term temporal information. 3) The discovered models achieve state-of-the-art performance on several video datasets and are among the fastest models for videos. We provide new diverse architectures, ensembles and components which can be reused for future work. To our knowledge this is the first automated neural architecture search algorithm for video understanding.

2. Related work

CNNs for video understanding. Approaches considering a video as a space-time volume have been particularly successful [1, 4, 30, 31], with a direct application of 3D CNNs to videos. C3D [30] learned 3x3x3 XYT filters, which was not only applied to action recognition but also to video object recognition. I3D [1] extended the Inception architecture to 3D, obtaining successful results on multiple activity recognition video datasets including Kinetics. S3D [38] investigated the usage of 1D and 2D convolutional layers in addition to the 3D layers. R(2+1)D [33] used the 2D conv. layers followed by 1D conv. layers while following the ResNet structure. Two-stream CNN design is also widely adopted in action recognition, which takes optical flow inputs in addition to raw RGBs [3, 27]. There are also works focusing on capturing longer temporal information in continuous videos using pooling [19], attention [20], and convolution [9]. Recurrent neural networks (e.g., LSTMs) are also used to sequentially represent videos [19, 39].

Neural architecture search. Neural network architectures have advanced significantly since the early convolutional neural network concepts of LeCun et al. [13] and Krizhevsky et al. [11]: from developing wider modules, e.g., Inception [28], or introducing duplicated modules [14], residual connections [5, 37], densely connected networks [6, 7], or multi-task architectures: e.g., Faster-RCNN and RetinaNet for detection, and many others [15, 16, 24]. Recently several ground-breaking approaches have been proposed for automated learning/searching of neural network architectures, rather than manually designing them [23, 29, 41, 42]. Successful architecture search has been demonstrated for images and text [41, 42], including object classification. Tran et al. [32] analyze action recognition experiments with different settings, e.g., input resolution, frame rate, number of frames, network depth, all within the 3D ResNet architecture.

3. Convolutional layers for action recognition

We first review standard convolutional layers for videos and then introduce the new iTGM layer to learn longer temporal structures with fewer parameters and lower computational cost. Video CNNs are analogous to standard CNNs, with the difference of an additional temporal dimension in the input and all intermediate feature maps. In more detail, both input and feature maps are represented as 4D tensors XYTC with two spatial dimensions, one temporal and one for the pixel values or features (i.e., channels). Several forms of convolution on such tensors have been explored.

3D convolutional layer learns a standard 3D convolutional kernel over space and time [8]. It applies C_{out} kernels of dimension $L \times H \times W \times C_{in}$ on a tensor of size $T \times Y \times X \times C_{in}$ to produce a tensor of size $T \times Y \times X \times C_{out}$. This layer has $LHWC_{in}C_{out}$ parameters, which is an order of magnitude larger than CNNs and becomes prohibitive in many cases. Further, expanding 2D kernels to 3D has been explored [17]. I3D expanded kernels by stacking the 2D kernels L times, results in state-of-the-art performance [1].

(2+1)D convolutional layer decomposes a 3D kernel into a composition of a 2D spatial kernel followed by a 1D temporal kernel [33, 38]. It has $HWC_{in}C_{out} + LC_{out}C_{out}$ parameters, and as such is more efficient than 3D convolution. However, it still depends on the time dimension L which limits the temporal size of the filter.

3.1. 3D Inflated TGM layer

The recently introduced Temporal Gaussian Mixture layer (TGM) [21] is a specialized 1D convolutional layer designed to overcome the limitations of standard 1D convolutional layers. In contrast to the standard 1D temporal convolutional layer, which was often used in video CNNs such as R(2+1)D, a TGM layer represents its filter as a mixture of 1D Gaussians. This makes the number of its learnable parameters independent of the temporal filter size; with a TGM layer, one does not have to handle all kernel weights but only the Gaussian mixture parameters.

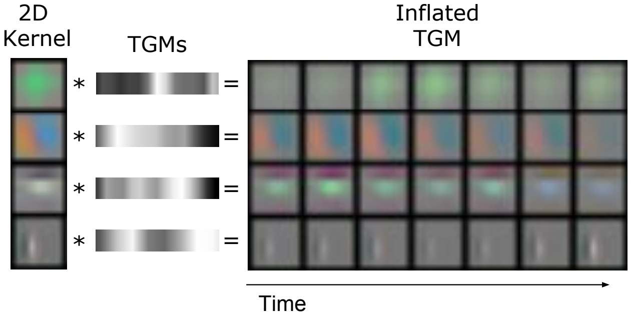

Figure 3. The iTGM layer. Example of inflated TGM kernels.

In this work, we employ the above idea to define a 3D space-time kernel directly, named *Inflated Temporal Gaussian Mixture* layer (iTGM). We 'inflate' the 2D spatial kernels to 3D by representing 3D kernel as a product of two kernels:

$$S \star K$$

where S is the 'inflated' 2D convolution and K is a temporal 1D kernel defined using a mixture of Gaussians (see Fig. 3).

The Gaussian mixture kernel K is defined as follows. Denote by μ_m and width σ_m the center and width of M Gaussians, $m \in \{0, \ldots, M\}$. Further, denote by a_{im}, $i \in \{0, \ldots, C_{out}\}$ soft-attention mixing-weights. The temporal Gaussian kernels read:

$$\hat{K}_{ml} = \frac{1}{Z} \exp\left(-\frac{(l-\mu_m)^2}{2\sigma_m^2}\right) \tag{1}$$

where Z is a normalization: $\sum_{l=0}^{L} \hat{K}_{ml} = 1$. Then, the a mixture of the above Gaussian kernels is:

$$K_{il} = \frac{\exp{(a_{im})}}{\sum_j \exp{(a_{ij})}} \hat{K}_{ml}. \tag{2}$$

This results in K being a $C_{out} \times L$ kernel; i. e., a temporal kernel with C_{out} output channels. We apply this kernel on the output of the spatial kernel. Thus, we obtain a $L \times H \times W \times C_{in} \times C_{out}$ kernel, using only $HWC_{in}C_{out} + 2M + MC_{out}$ parameters.

In practice, μ is constrained to be in $[0, L)$, $\mu = (1/2)(L-1)\tanh{(\hat{\mu})} + 1$. and σ is positive, $\sigma^2 = \exp{(\hat{\sigma})}$. Further, M is a hyperparameter, typically smaller than L.

The parameters of the iTGM layer – spatial kernel parameters, μ_m, σ_m, and a_{im} – are all differentiable, and are learned from data for the specified task. The above layer behaves exactly like the standard 3D XYT convolution. Note that this layer learns fewer parameters than both 3D and (2+1)D convolution, and can learn temporally longer kernels as the number of parameters is independent of the length, L. Examples of inflated TGMs are shown in Fig. 3.

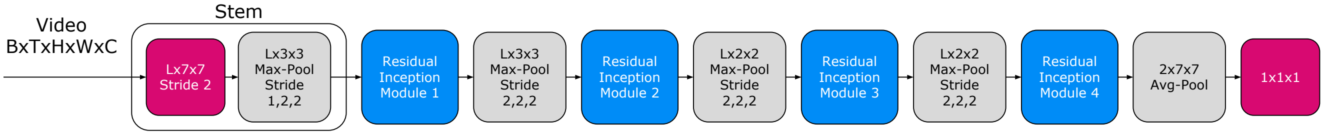

Figure 4. Our ResNet-like 'fill-in-the-blanks' meta-architecture: each heterogeneous module is repeated R times, based on the evolution.

4. Neural architecture evolution for videos

We design our neural architecture search specifically for videos, and propose the following:

- Use of 'fill-in-the-blanks' meta-architectures to limit the search space and generate both trainable and high-performing architectures.
- Search among combinations of six different types of space-time convolution/pooling layer concatenations where their temporal duration can vary in large ranges.
- We specially design mutation operations to more effectively explore the large space of possible architectures.
- We propose an evolutionary sampling strategy which encourages more diverse architectures early in the search.

Neural architecture evolution finds better-performing architectures by iteratively modifying a pool of architectures. Starting from a set of random architectures, it mutates them over multiple rounds, while only retaining the better performing ones. Recent studies [22] show that evolutionary algorithms can find good image architectures from a smaller number of samples, as opposed to model search algorithms using reinforcement learning [41]. This makes evolution more suitable for video architecture search, as video CNNs are expensive to train. Further, it allows for mutating architectures by selecting and combining various space-time layers which more effectively process inputs with much larger dimensionality. The evolution also enables obtaining multiple different architectures instead of a single architecture which we use to build a powerful ensemble.

4.1. Search space and base architecture

We evolve our architectures to have heterogeneous modules, motivated by the recent observations that video CNN architectures may need differently sized temporal filters at different layers, e.g., bottom-heavy vs. top-heavy [38]. In order to keep the entire search space manageable while evolving modules heterogeneously, we use a meta-architecture where internal sub-modules are allowed to evolve without constraints but the high level architecture has a fixed number of total modules. We used both an Inception-like and ResNet-like meta-architecture. The Inception meta-architecture follows the popular Inception architecture, with five layers forming the 'stem' followed by Inception modules whose structure is evolved. The ResNet-like meta-architecture is illustrated in Figure 4. This meta-architecture is composed of two fixed convolutional layers (i.e., the 'stem') followed by four residual Inception modules interspersed with max-pooling layers. Each residual Inception module can be repeated R times and has a residual connection. Figure 5 shows an example module.

Each module can have multiple parallel convolutional or pooling layers and its specific form is chosen through evolution. We constrain the complexity of the connections between the layers within a module while making the evolution explore temporal aspects of the modules. More specifically, we make each module have 1-6 parallel 'streams' with four different stream types: (i) one 1x1x1 conv., (ii) one space-time conv. layer after one 1x1x1 layer, (iii) two space-time conv. layers after one 1x1x1, and (iv) a space-time pooling followed by one 1x1x1. Figure 5 shows the four types. The architecture evolution focuses on modifying each module: selecting layer types and its parameters, selecting the number of parallel layers, and for the residual ones, how many times should each module be repeated.

The convolutional layers have $\{1, 3, 5, 7, 9, 11\}$ as the set of possible temporal kernel sizes. As a result, the architecture search space size is $O((3\times6+1)^{5+B\times N}+(6+1)^{D\times N})$ where B and D are the maximum of number of space-time conv and pooling layers we allow in each module, and $N = 4$ or 9 is the number modules in the meta-architecture. There are 2 or 5 individual layers (often also called a 'stem') before the modules. Each space-time conv. layer has 3×6 possible options and each space-time pooling has 6 options. Also, there is the option to add/omit the layer, making the total number of choices $3\times6+1$ and $6+1$. For the ResNet-like models, we allow modules to be repeated up to 6 times. We fix the spatial size of the kernels to be 3×3. Although the search space is very big, the idea is that an exhaustive search is not necessary and it is possible to find good local optima by evolving from various initial samples (i.e., architectures).

4.2. Evolutionary algorithm

Algorithm 1 summarizes the architecture search. In a standard genetic algorithm setting, we maintain a population of size P, where each individual in the population is a particular architecture. Initial architectures are obtained by randomly sampling from our large search space, encouraging diversity and exploration. At each round of the evolution, the algorithm randomly selects S number of samples from the entire population and compares their recognition performance. The architecture with the highest fitness (i.e., validation accuracy) becomes the 'parent', and mutation operators are applied to the selected parent to generate a new

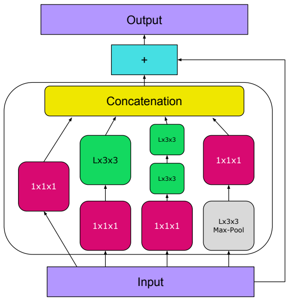

Figure 5. A example structure of the a residual Inception module with 4 layer streams. There could be 1-6 parallel streams (with 4 types) and a residual connection from input to output.

‘child’ architecture to be added to the population. Whenever a new architecture is added, it is trained with the training set for a number of iterations, and is evaluated with a separate validation set (different from the actual test and validation sets) to measure the recognition accuracy. This performance becomes the ‘fitness’ of the architecture. Having S where $1 < S \leq P$ controls the randomness in of the parent selection. It avoids the algorithm repeatedly selecting the same parent, which might already be at a local maximum.

Algorithm 1 Evolutionary search algorithm

```
function SEARCH
    Randomly initialize the population, $P$
    Evaluate each individual in $P$
    for $i <$ number of evolutionary rounds do
        $S =$ random sample of 25 individuals
        $parent =$ the most fit individual in $S$
        $child = parent$
        for $\max(\lceil d - \frac{i}{r} \rceil, 1)$ do
            $child = mutate(child)$
        end for
        evaluate $child$ and add to population
        remove least fit individual from population
    end for
end function
```

Mutations. The mutation operators modify the parent architecture to generate a new child architecture. In order to explore the architecture search space we describe in Section 4.1 efficiently, we consider the following 4 mutation operators: (i) Select a space-time conv. layer within the parent architecture, and change its ‘type’. (ii) Select a space-time conv. layer or a pooling layer, and change its temporal size (i.e., L) . (iii) Select a module from the parent architecture, and add/remove a parallel layer stream. We constrain the number of parallel layer streams to be 1-6. We additionally constrain each module to have a fixed number of output filters which are evenly divided between the parallel layers. (iv) Select a module and change the number of times it is repeated. Figure 6 illustrates examples of our mutation operators applied to layers of a module.

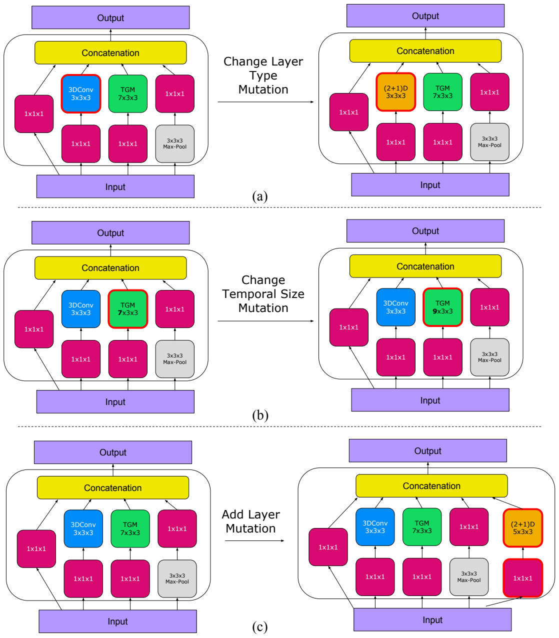

Figure 6. Example mutations applied to a module, including (a) layer type change, (b) filter length change, and (c) layer addition.

Diversity. Importantly, we design the mutation in our algorithm to happen by applying multiple randomly chosen mutation operators. In order to encourage more diverse architectures, we develop the strategy of applying many mutation operators in the early stage of the evolution while reducing the amount of mutations in the later stages, which is analogous to controlling the learning rate in a CNN model learning. As described in Algorithm 1, we apply $\max(d - \frac{i}{r}, 1)$ number of mutation operators where d is the maximum number of operators we want to apply in the beginning, and r controls how quickly we want to decrease their numbers linearly. Once a child architecture is added to the population, in order to maintain the size of the population to P, the evolutionary algorithm selects an individual to discard from the pool. We tried different removal criteria including the lowest fitness and the oldest (i.e., [22]), which did not make much difference in our case.

Ensemble. We obtain a number of top performing architectures after the evolutionary search is completed, thanks to our evolutionary algorithm promoting populations with diverse individual architectures. Thus, we are able to construct a strong ensemble from the diverse models by averaging the outputs of their softmax layers: $F^*(x) = \sum_i F_i(x)$ where x is the input video and F_i are the top models. In the experiments, we found our approach obtains very diverse, top performing architectures. Ensembling further improves the overall recognition. We named our final ensemble network as EvaNet (Evolved Video Architecture).

5. Experiments

Although our evolutionary architecture search is applicable to various different video understanding tasks, here we focus on human activity recognition. The video CNN architectures are evolved using public datasets. Fitness of the architectures during evolution is measured on a subset of the training data. In all experiments, the evolutionary algorithm has no access to the test set during training and evolution. In more detail, we use following datasets:

HMDB [12] is a dataset of human motion videos collected from a variety of sources. It is a common datasets for video classification and has ∼7K videos of 51 action classes.

Kinetics [10] is a large challenging video dataset with 225,946 training and 18,584 validation videos. We use the currently available version (Kinetics-400 dataset), which has about 25k fewer training videos than original Kinetics dataset (i.e., missing about 10% of train/val/test data). This makes the dataset more difficult to train, and not comparable to the previous version.

Charades [26] is an activity recognition dataset with ∼10K videos, whose durations are 30 seconds on average. We chose Charades to particularly confirm whether our architecture evolution finds structures different from those found with shorter videos like Kinetics. We use the standard classification evaluation protocol.

Moments in Time [18] is a large-scale dataset for understanding of actions and events in videos (339 classes, 802,264 training, 33,900 validation videos).

5.1. Experimental setup

Architecture evolution is done in parallel on smaller input size and fewer number of iterations. Details can be found in the appendix. We perform evolution for 2000 rounds: generating, mutating, training/evaluating, and discarding 2000 CNN architectures. Note that ∼300 rounds were often sufficient to find good architectures (Figure 7). Once the architecture evolution is complete and the top performing models are found, they are trained on full inputs.

Baselines. We compare our results to state-of-the-art activity recognition methods. We train (1) the original I3D [1] with standard 3D conv. layers. We also train an Inception model with: (2) 3D conv. layers with $L = 3$, (3) (2+1)D conv. layers, and (4) the proposed iTGM layers. The difference between (1) and (2) is that (1) uses $L = 7$ in the first 3D conv. layer and $L = 3$ in all the other 3D layers (a handcrafted design), while (2) uses $L = 3$ in all its layers.

5.2. Results

Next, we report the results of the proposed method and compare with baselines and prior work. This is not only done in terms of recognition accuracy but also in terms of computational efficiency. As shown in Table 7, our individual models are 4x faster and the ensemble (EvaNet) is 1.6x faster than standard methods like ResNet-50. Both of our meta-architectures perform similarly. Below, we report results of the ResNet-like architecture (see suppl. material for further results).

Table 1. HMDB split 1 comparison to baselines, with and without Kinetics pre-training. The models were all initialized with ImageNet weights.

	HMDB			HMDB(pre-train)		
	RGB	Flow	RGB+F	RGB	Flow	RGB+F
Baselines						
I3D	49.5	61.9	66.4	74.8	77.1	80.1
3D Conv	47.4	60.5	65.9	74.3	76.8	79.9
(2+1)D Conv	27.8	56.4	51.8	74.4	76.5	79.9
iTGM Conv	56.5	62.5	68.2	74.6	76.7	79.9
3D-Ensemble			67.6			80.4
iTGM-Ensemble			69.5			80.6
Top individual models from evolution						
Top 1	60.7	63.2	70.3	74.4	78.7	**81.4**
Top 2	63.4	62.5	**71.2**	75.8	78.4	80.6
Top 3	60.5	63.1	70.5	75.4	78.9	79.7
EvaNet			**72.8**			**82.7**

Table 2. HMDB performances averaged over the 3 splits.

Two-stream [27]	59.4
Two-stream+IDT [3]	69.2
R(2+1)D [33]	78.7
Two-stream I3D [1]	80.9
PoTion [2]	80.9
Dicrim. Pooling [35]	81.3
DSP [34]	81.5
Top model (Individual, ours)	81.3
3D-Ensemble	79.9
iTGM-Ensemble	80.1
EvaNet (Ensemble, ours)	**82.3**

HMDB: Table 1 shows the accuracy of the evolved CNNs compared to the baseline architectures, where the evaluation is done on 'split 1'. We see improved accuracy of our individual models as well as ensembles. We also confirm that the EvaNet ensemble is superior to the ensembles obtained by combining other architectures (e.g., 3D ResNet). Table 2 compares our performance with the previous state-of-the-arts on all three splits following the standard protocols. As seen, our EvaNet models have strong performances outperfoming the state-of-the-art.

Kinetics: Table 3 shows the classification accuracy of our algorithm on Kinetics-400, and compares with baselines, other ensembles, and the state-of-the-art. The architecture evolution finds better performing models than any prior model. Further the ensemble of 3 models (EvaNet) improves the performance and outperforms other ensembles, including and ensemble of diverse, standard architectures.

Table 3. Performances on Kinetics-400 Nov. 2018 version. Note that this set is ∼10% smaller (in training/validation set size) than the initial version of Kinetics-400. We report the numbers based on models trained on this newest version. Baselines are shown on top, followed by the state-of-the-arts, and then our methods.

Method	Accuracy
3D Conv	72.6
(2+1)D Conv	74.3
iTGM Conv	74.4
ResNet-50 (2+1)D	72.1
ResNet-101 (2+1)D	72.8
3D-Ensemble	74.6
iTGM-Ensemble	74.7
Diverse Ensemble (3D, (2+1)D, iTGM)	75.3
Two-stream I3D [1]	72.6
Two-stream S3D-G [38]	76.2
ResNet-50 + Non-local[36]	73.5
Arch. Ensemble (I3D, ResNet-50, ResNet-101)	75.4
Top 1 (Individual, ours)	**76.4**
Top 2 (Individual, ours)	75.5
Top 3 (Individual, ours)	75.7
Random Ensemble	72.6
EvaNet (Ensemble, ours)	**77.2**

Table 4. Charades classification results against state-of-the-arts.

	mAP
Two-Stream [25]	18.6
Two-Stream + LSTM [25]	17.8
Async-TF [25]	22.4
TRN [40]	25.2
Dicrim. Pooling [35]	26.7
Non-local NN [36]	37.5
3D-Ensemble (baseline)	35.2
iTGM-Ensemble (baseline)	35.7
Top 1 (Individual, ours)	**37.3**
Top 2 (Individual, ours)	36.8
Top 3 (Individual, ours)	36.6
EvaNet (Ensemble, ours)	**38.1**

Charades: We also test our approach on the popular Charades dataset. Table 4 compares against the previously reported results (we use Kinetics pre-training as in [36]). As shown, we outperform the state-of-the-art and establish a new one with our EvaNet. Our CNNs only use RGB input (i.e., one-stream) in this experiment.

Transfer learned architectures - Moments in Time: We evaluate the models evolved on Kinetics by training it on another dataset: Moments in Time [18]. Table 5 shows the results, where we see that the models outperform prior methods and baselines. This is particularly appealing as the evolution is done on another dataset and successfully transfers to a new dataset.

Table 5. Moments in time. We show that models evolved on Kinetics transfer to similar datasets.

Method	Accuracy
I3D [18]	29.5
ResNet-50	30.5
ResNet-50 + NL [36]	30.7
Arch. Ensemble (I3D, ResNet-50, ResNet-101)	30.9
Top 1 (Individual, ours)	**30.5**
EvaNet (Ensemble, ours)	**31.8**

Table 6. Test accuracy across datasets for a model evolved on a single dataset.

Method	Kinetics	Charades	HMDB	MiT
Evolved on Kinetics	**77.2**	37.8	**82.3**	**31.8**
Evolved on Charades	76.5	**38.1**	81.8	31.1
Evolved on HMDB	77.0	37.5	**82.3**	31.6
Best without evolution	76.2	37.5	81.5	30.7

Table 7. Runtime measured on a V100 GPU. Accuracy numbers on Kinetics-400 are added for context. These numbers are evaluation time for 1 128 frame clip at 224x224.

Method	Accuracy	Runtime
I3D	72.6	337ms
S3D	75.2	439ms
ResNet-50	71.9	526ms
ResNet-50 + Non-local	73.5	572ms
I3D iTGM (ours)	74.4	274ms
Individual learned model (ours)	75.5	**108ms**
EvaNet (Ensemble, ours)	77.2	**258ms**

Ensembling and runtime. One key benefit of evolving model architectures is that the resulting models are naturally diverse, as they are evolved from very different initial random models. As shown in Table 3, we compared with an ensemble of three different baselines (3D Conv + (2+1)D + iTGM) and with an ensemble of different architectures (e.g., I3D + ResNet-50 + ResNet-101). Both are outperformed by EvaNet, although the base models are individually strong.

Furthermore, our evolved models are very efficient performing inference on a video in ∼100 ms (Table 7). Note that even an ensemble is faster, 258 ms, than previous individual models which makes the proposed approach very suitable for practical use with higher accuracy and faster runtimes. This gain in runtime is due to the use of parallel shallower layers and the use of iTGM layers, which is by itself faster than prior layers (274ms vs 337ms).

Architecture findings. Figures 1 and 2 show examples of the architectures found. Interesting substructures discovered include: (1) modules combining multiple space-time pooling layers with different temporal intervals and (2)

Table 8. Comparison between models from different hybrid meta-architectures. Kinetics dataset.

Method	Accuracy
EvaNet Inception (Ensemble, ours)	76.8
EvaNet ResNet (Ensemble, ours)	77.2
EvaNet Combined (Ensemble, ours)	**77.4**

Table 9. Statistics of the top models. iTGM layers are most common and have longest temporal duration. Kinetics dataset.

	Number of Layers			Ave. Temporal Length			
	3D	(2+1)D	iTGM	3D	(2+1)D	iTGM	Pool
Top 1	2	6	16	5	7.2	7.2	6.0
Top 2	6	7	12	7.8	8.1	8.6	5.7
Top 3	2	6	15	6	7.8	8.5	6.2

modules heavily relying on Inflated TGM or (2+1)D conv. layers instead of standard 3D conv. layers. Such modules were commonly observed at most of the locations in the architectures, while being very diverse and heterogeneous.

Video CNN architectures may evolve differently depending on the datasets. This is as expected, and we were able to explicitly confirm this. The architectures have many more layers with longer space-time filters (e.g., 9 or 11) when evolved for Charades, while they only had a small number of them when evolved for HMDB or Kinetics. An average activity duration in Charades videos are around 12 seconds, while HMDB and Kinetics videos are on the average of 3 to 5 seconds. Different architectures are needed for different datasets/tasks, and we are providing an evolutionary approach to automate the architecture design.

Table 8 further shows that both Inception-like and ResNet-like meta-architectures are successful, and a combination of them is even more successful.

5.3. Ablation Studies

Effectiveness of iTGM models. In Table 9, we show the layer statistics for the best models. In the EvaNet architecture, iTGM layers have the longest average length (8.6). Further, our models have quite large temporal resolution of 368 frames on average (compared to I3D/S3D with 99 frames). To further confirm the usefulness of the iTGM layer, we conduct several experiments. In Table 10, we show the results using iTGM layers with various temporal durations. Since we can increase the temporal length without changing the number of parameters, we can improve performance by simply taking longer temporal durations. We also compare to replacing all iTGM layers with (2+1)D layers and performing the architecture search without the iTGM layer as an option. Both restrictions degrade performance, confirming that iTGMs are needed. We also note that iTGM layers are most common in the best models (Table 9), further confirming their importance.

'Stretching' of iTGM layer Since the number of parameters of the iTGM layer is independent of length, we use a model from the Kinetics dataset and 'stretch' the iTGM layers and apply it to Charades, which has activities with much longer temporal duration. In Table 11, we show the results using models with $L = 3$ on Kinetics and stretched to $L = 11$ on Charades, which shows similar performance.

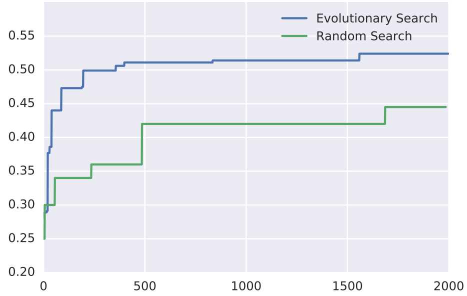

Figure 7. Random search vs. evolutionary algorithm on HMDB. X axis is number of rounds, Y axis is accuracy.

Table 10. Experiments evaluating effect of iTGM layer on Kinetics.

Model	Accuracy
iTGM ($L = 3$)	74.4
iTGM ($L = 11$)	74.9
EvaNet replacing iTGM with (2+1)D	76.6
Arch Search without iTGM in space	76.8
EvaNet	**77.2**

Table 11. Stretching iTGM kernels from Kinetics to Charades.

Model	mAP
iTGM Baseline ($L = 3$)	33.8
iTGM Stretched ($L = 11$)	34.2
Kinetics EvaNet	37.7
Kinetics EvaNet Stretched ($L = 11$)	**38.1**
Charades EvaNet	**38.1**

Evolution vs. random search. We compared our architecture evolution with random architecture search (Figure 7). We observe that both the evolution and the random search accuracies improve as they explore more samples (benefiting from the search space designed). However, the architecture evolution obtains much higher accuracy and much more quickly with few initial rounds of evolution, suggesting the mutations are being effective.

6. Conclusion

We present a novel evolutionary algorithm that automatically constructs architectures of layers exploring space-time interactions for videos. The discovered architectures are accurate, diverse and very efficient. Ensembling such models leads to further accuracy gains and yields faster and more accurate solutions than previous state-of-the-art models. Evolved models can be used across datasets and to build more powerful models for video understanding.

References

[1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 3, 6, 7

[2] Vasileios Choutas, Philippe Weinzaepfel, Jérôme Revaud, and Cordelia Schmid. Potion: Pose motion representation for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[3] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016. 1, 2, 6

[4] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the ICCV Workshop on Action, Gesture, and Emotion Recognition*, volume 2, page 4, 2017. 2

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 3

[6] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[7] Simon Jegou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. One hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. 2016. 3

[8] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. In *International Conference on Machine Learning (ICML)*, pages 495–502, 2010. 3

[9] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014. 1, 2

[10] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 6

[11] A Krizhevsky, I Sutskever, and GE Hinton. Imagenet classification with deep convolutional neural networks. 2012. 3

[12] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2011. 6

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. 1998. 3

[14] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. 2013. 3

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. 2017. 3

[16] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. 2018. 3

[17] Elman Mansimov, Nitish Srivastava, and Ruslan Salakhutdinov. Initialization strategies of spatio-temporal convolutional neural networks. *arXiv preprint arXiv:1503.07274*, 2015. 3

[18] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfruend, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 6, 7

[19] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702. IEEE, 2015. 2, 3

[20] AJ Piergiovanni, Chenyou Fan, and Michael S Ryoo. Learning latent sub-events in activity videos using temporal attention filters. In *Proceedings of the American Association for Artificial Intelligence (AAAI)*, 2017. 2

[21] AJ Piergiovanni and Michael S. Ryoo. Temporal gaussian mixture layer for videos. In *International Conference on Machine Learning (ICML)*, 2019. 1, 3

[22] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018. 4, 5

[23] Esteban Real, Sherry Moore, Andrew Selle, Yutaka Leon Suematsu Saurabh Saxena, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning (ICML)*, 2017. 3

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. 2015. 3

[25] Gunnar A Sigurdsson, Santosh Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. *arXiv preprint arXiv:1612.06371*, 2016. 7

[26] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 6

[27] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014. 2, 6

[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. 1, 3

[29] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*, 2018. 1, 3

[30] Du Tran, Lubomir D Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3d: generic features for video analysis. *CoRR, abs/1412.0767*, 2(7):8, 2014. 1, 2

[31] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017. 2

[32] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017. 3

[33] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 3, 6

[34] Jue Wang and Anoop Cherian. Learning discriminative video representations using adversarial perturbations. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2018. 6

[35] Jue Wang, Anoop Cherian, Fatih Porikli, and Stephen Gould. Video representation learning using discriminative pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1149–1158, 2018. 6, 7

[36] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 7

[37] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[38] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding, 2018. 1, 2, 3, 4, 7

[39] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision (IJCV)*, pages 1–15, 2015. 3

[40] Bolei Zhou, Alex Andonian, and Antonio Torralba. Temporal relational reasoning in videos. *arXiv preprint arXiv:1711.08496*, 2017. 7

[41] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 3, 4

[42] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3