SlowFast Networks for Video Recognition

- This algorithm is from Facebook AI research
- PySlowFast: high performance, light weight and efficient video understanding codebase in pytorch.
- Slowfast AI algorithm recognized what activity is being performed in the video. It also can detect any action is happening
- Used for video understanding research on different tasks (classification, detection, & etc.).
- **SlowFast:** novel method to analyze the contents of a video segment.
- It has two path away, one is capturing semantics of the object which is class slow path away another one is fast path away which capture the motion.
- Both paths performs 3D convolution operation
- **Github:** https://github.com/facebookresearch/SlowFast

Backbone model architecture:

- SlowFast
- Slow
- C2D
- I3D
- Non-local Network
- X3D
- MViTv1 and MViTv2

Kinetics Dataset

- **Kinetics**: datasets of URL links of up to 650,000 video clips that cover 400/600/700 human action classes, depending on the dataset version
- **Videos include:** human-object interactions, human-human interactions such as shaking hands and hugging
- Each action class has at least 400/600/700 video clips
- Each clip is human annotated with a single action class and lasts around 10 seconds.
- Load a pre trained video classification model in PyTorchVideo and run it on a test video
- Running SlowFast networks pre-trained on the Kinetics 400 dataset.
- **Link:** https://www.deepmind.com/open-source/kinetics

SlowFast networks pre-trained on the Kinetics 400 dataset

- Create virtual environment on conda and run on jupyter / google colab notebook
- Install all required libraries and packages **https://github.com/facebookresearch/SlowFast/blob/main/INSTALL.md**
- **Perform the code:**

1. Import all functions
2. Load the model
3. Set the model to eval mode and move to desired device
4. Download the id to label mapping for the Kinetics 400 dataset on which the torch hub models were trained.
5. This will be used to get the category label names from the predicted class ids.
6. Define input transform
7. Before passing the video into the model we need to apply some input transforms and sample a clip of the correct duration
8. Run Inference
9. Download an example video
10. Load the video and transform it to the input format required by the model
11. Get Predictions

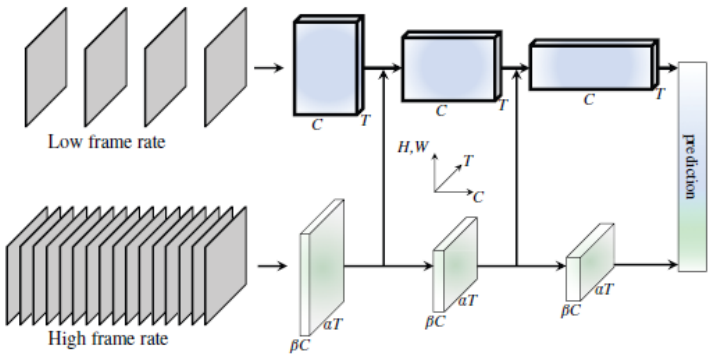
SlowFast Model by FAIR PyTorchVideo

- SlowFast model architectures are based on [1] with pre-trained weights using the 8x8 setting on the Kinetics dataset.
- Both the Slow and Fast pathways use a 3D ResNet model, capturing several frames at once and running 3D convolution operations on them

arch	depth	frame length x sample rate	top 1	top 5	Flops (G)	Params (M)
SlowFast	R50	8x8	76.94	92.69	65.71	34.57
SlowFast	R101	8x8	77.90	93.27	127.20	62.83

Reference: [1] Christoph Feichtenhofer et al, "SlowFast Networks for Video Recognition" https://arxiv.org/pdf/1812.03982.pdf

Slow pathway and Fast pathway parameters



Model Implementation: Google Colab Notebook Code Link Given Below:

Link: https://colab.research.google.com/drive/1xw0SCaCRTN-KaPAZeLY_lPRnmctdPSNx

Sample video:

Results:

```
# Pass the input clip through the model
preds = model(inputs)

# Get the predicted classes
post_act = torch.nn.Softmax(dim=1)
preds = post_act(preds)
pred_classes = preds.topk(k=5).indices[0]

# Map the predicted classes to the label names
pred_class_names = [kinetics_id_to_classname[int(i)] for i in pred_classes]
print("Top 5 predicted labels: %s" % ", ".join(pred_class_names))
```

```
Top 5 predicted labels: archery, throwing axe, playing paintball, disc golfing, riding or walking with horse
```

3D RESNET on Slow Architecture Network

- The model architecture is based on [1] with pre-trained weights using the 8x8 setting on the Kinetics dataset.

Reference:

[1] Christoph Feichtenhofer et al, “SlowFast Networks for Video Recognition” https://arxiv.org/pdf/1812.03982.pdf

arch	depth	frame length x sample rate	top 1	top 5	Flops (G)	Params (M)
Slow	R50	8x8	74.58	91.63	54.52	32.45

- A residual neural network (ResNet) is **an artificial neural network (ANN)**
- ResNet 3D is **a type of model for video that employs 3D convolutions**.
- This model collection consists of two main variants.
- The first formulation is named mixed convolution (MC) and consists in employing 3D convolutions only in the early layers of the network, with 2D convolutions in the top layers.

Model Implementation: Google Colab Notebook Code Link Given Below:

Link: https://colab.research.google.com/drive/1NKx1RH8sZzJQ88T3_dTDbWKIZjSFOlYF

Sample video:

Results:

```
[ ]  # Pass the input clip through the model
     preds = model(inputs[None, ...])

     # Get the predicted classes
     post_act = torch.nn.Softmax(dim=1)
     preds = post_act(preds)
     pred_classes = preds.topk(k=5).indices[0]

     # Map the predicted classes to the label names
     pred_class_names = [kinetics_id_to_classname[int(i)] for i in pred_classes]
     print("Top 5 predicted labels: %s" % ", ".join(pred_class_names))
```

```
Top 5 predicted labels: archery, throwing axe, playing paintball, stretching arm, riding or walking with horse
```

X3D Model by FAIR PyTorchVideo

- X3D model architectures are based on pre-trained on the Kinetics dataset

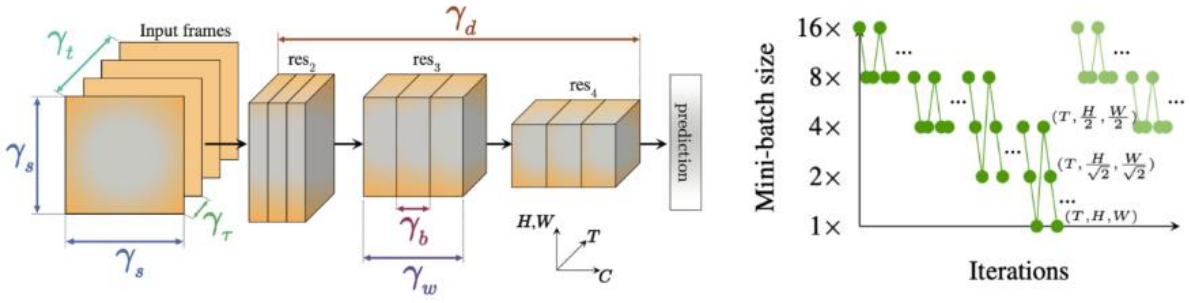
arch	depth	frame length x sample rate	top 1	top 5	Flops (G)	Params (M)
X3D	XS	4x12	69.12	88.63	0.91	3.79
X3D	S	13x6	73.33	91.27	2.96	3.79
X3D	M	16x5	75.94	92.72	6.72	3.79

Reference:

[1] Christoph Feichtenhofer, "X3D: Expanding Architectures for Efficient Video Recognition." https://arxiv.org/abs/2004.04730

X3D: Expanding Architectures for Efficient Video Recognition

- X3D is Multi-grid Training:
- Multi-grid training is a mechanism to train video architectures efficiently. Instead of using a fixed batch size for training, this method proposes to use varying batch sizes in a defined schedule, yet keeping the computational budget approximately unchanged by keeping batch x time x height x width a constant.



Model Implementation: Google Colab Notebook Code Link Given Below:

Link: https://colab.research.google.com/drive/1rd5c0H9dzL5C7ltVQ2IPEa1aRZUEvF8D#scrollTo=rVa8kxbN9-aP

Sample Video:

Predicted Results:

```
Archery, air drumming, applauding, applying cream, abseiling
```

```
# Pass the input clip through the model
preds = model(inputs[None, ...])

# Get the predicted classes
post_act = torch.nn.Softmax(dim=1)
preds = post_act(preds)
pred_classes = preds.topk(k=5).indices[0]

# Map the predicted classes to the label names
pred_class_names = [kinetics_id_to_classname[int(i)] for i in pred_classes]
print("Top 5 predicted labels: %s" % ", ".join(pred_class_names))
```

```
Top 5 predicted labels: archery, air drumming, applauding, applying cream, abseiling
```

Results comparison:

	Top-1 result	Top-5 result
Slow-Only	72.6%	90.3%
Fast-Only	51.7%	78.5%
Previous State-of-the-art	73.9%	90.9%
SlowFast	79.0%	93.6%