API GUIDE

1. UPLOAD MODEL

To upload a model the following specifications must be met.
The checkpoint file must contain the following fields

“arch” - The architecture of pretrained model used by your model. For example densenet121. See the list of supported architectures below

- Densenet121
- resnet18 = models.resnet18(pretrained=True)
- alexnet = models.alexnet(pretrained=True)
- squeezenet = models.squeezenet1_0(pretrained=True)
- vgg16 = models.vgg16(pretrained=True)
- densenet = models.densenet161(pretrained=True)
- inception = models.inception_v3(pretrained=True)

“Network” - The structure of the network model in JSON format.

For example, a classifier like this

```
classifier = nn.Sequential(OrderedDict([
        ('fc1', nn.Linear(1024, 300)),
        ('relu1', nn.ReLU()),
        (‘drop1’ , nn.Dropout(0.2)),
        ('fc2', nn.Linear(300, 102)),
        ('output', nn.LogSoftmax(dim=1))]))
```

Corresponds to the following json fomat

```
data ={
  "input":{
    "name":"fc1",
    "type":"linear",
    "in":1024,
    "out":300,
    "act":"relu",
    "drop":0.2,
    "actname":"relu1",
    "dropname":"drop1"
  },
  "1":{
    "name":"fc2",
    "type":"linear",
    "in":300,
    "out":102
  },
  "output":{
    "name":"output",
    "type":"logsoft",
    "dim":1
  }
}
```

“**state_dict**”: --
“**Class_to_idx**”: --

Response:

```
{“status”: “saved”
“model_id“:”830388996299723”
}
```

2. PREDICT:

Request

```
{
“File: --
}
```

Response:

```
{
  “name”: --
}
```

Currently, the predict function works with a static model defined in the backend. There is a new predict API called “predict_new” which expects the standard of the upload to be respected in order to make predictions. Predictions are made using 3 random available models in the model store. The old predictions functionality will, however, be depreciated.

3. PREDICT WITH A MODEL

Request

```
{
“file: --,
“model_id”:””
}
```

Response:

```
{
  “name”: --
}
```