**УТВЕРЖДАЮ**

Должность

\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО

“ ” \_\_\_\_\_\_\_\_\_ 2018 г.

**Пояснительная записка 5.**

**Технология обучения нейронной сети**

ОКР **«**Построение и обучение нейронной сети для идентификации фокусного расстояния по серии изображений**»   
(«**SmartGetDistance**»)**

Н.Новгород

2018

Реферат

Пояснительная записка 29 страниц, 2 источника.

В пояснительной записке в рамках разработки проекта описан процесс обучения нейронной сети NVIDIA DIGITS на примере нескольких классов

Оглавление

[Термины и определения 4](#_Toc531948065)

[1 Технология обучения нейронной сети по трем классам 5](#_Toc531948066)

[1.1 Формирование данных 5](#_Toc531948067)

[1.2 Формирование модели 13](#_Toc531948068)

[2 Технология дообучения уже существующей модели нейронной сети 20](#_Toc531948069)

[2. 1. Выгрузка модели 20](#_Toc531948070)

[2. 2. Дообучение существующей модели 21](#_Toc531948071)

[3 Тестирование модели 24](#_Toc531948072)

[Список используемых источников 29](#_Toc531948073)

# Термины и определения

**ПО «SmartGetDistance»** - разрабатываемое в рамках текущей ОКР ПО предназначенное для построения и обучения нейронной сети для определения расстояния между фокусом оптической системы и поверхностью рельефа по серии изображений.

**DIGITS** (Deep Learning GPU Training System) - набор программ для создания глубинных нейронных сетей (DNN) в ходе машинного обучения, а также для управления и диагностики данного процесса. DIGITS обладает графическим пользовательским интерфейсом. В состав пакета входит веб-сервер, с помощью которого осуществляется коллективная работа над проектом.

**Python** — [высокоуровневый язык программирования](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)общего назначения, ориентированный на повышение производительности разработчика и читаемости кода.

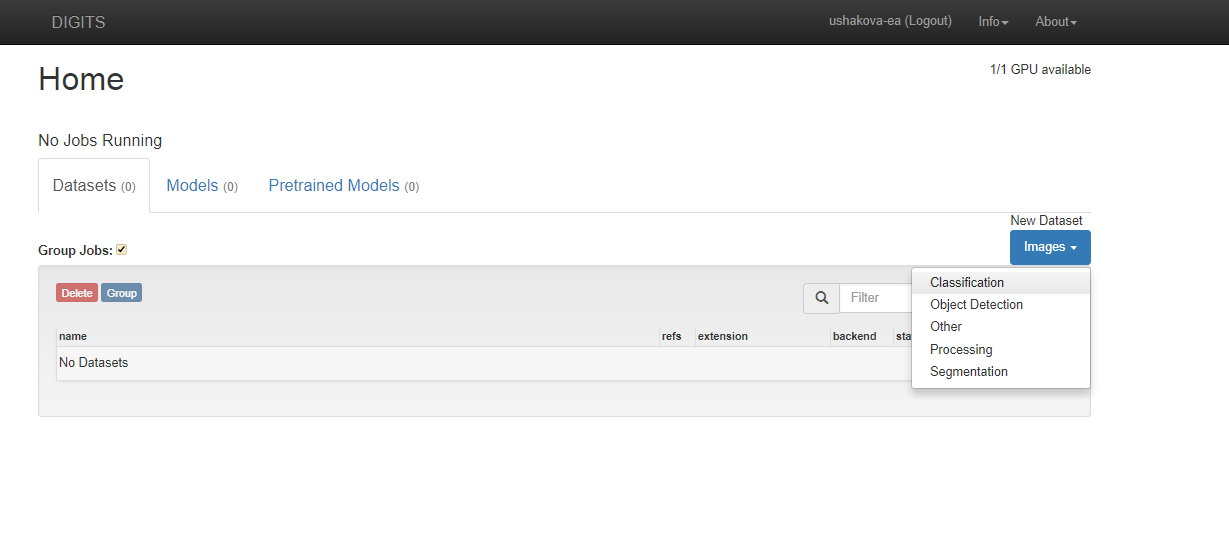
**Эпоха** - одна итерация в процессе обучения, включающая предъявление всех примеров из обучающего множества и, возможно, проверку качества обучения на контрольном множестве.

# Технология обучения нейронной сети по трем классам

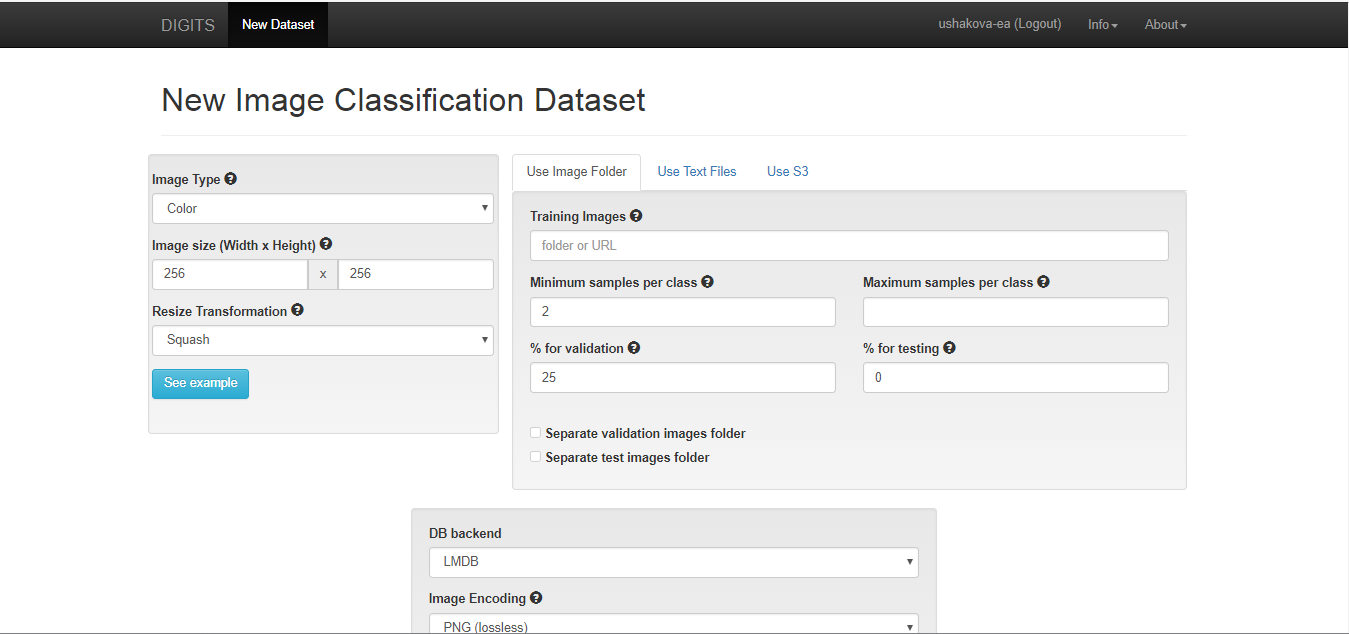
На машине необходимо развернуть нейронную сеть NVIDIA DIGITS (см. Инструкцию по развертыванию нейронной сети NVIDIA DIGITS).

## Формирование данных

1. Для формирования данных для решения задачи классификации необходимо открыть вкладку Datasets и выбрать Classification.



1. Далее откроется форма, в ней выбрать вкладку Use Image Folder

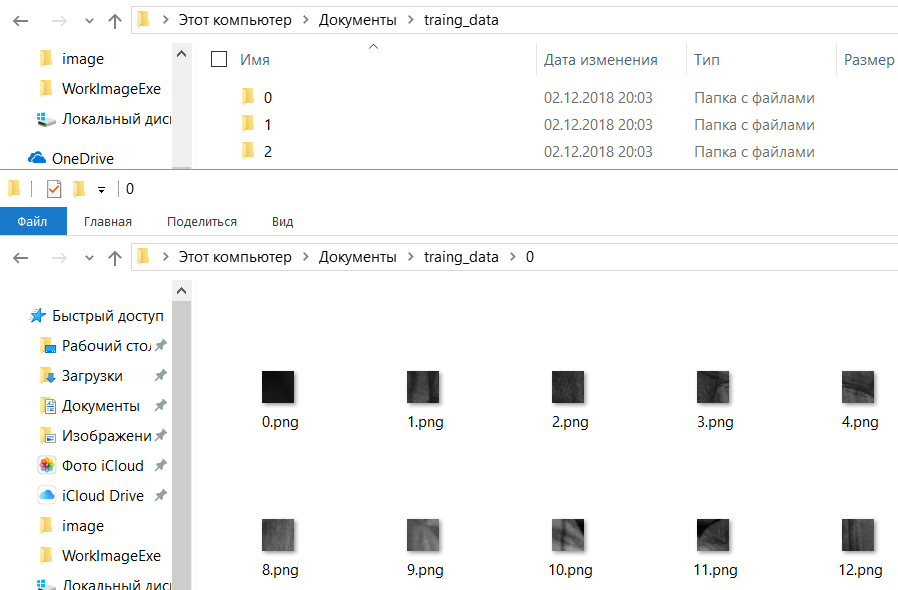


1. В форме присутствуют следующие параметры:

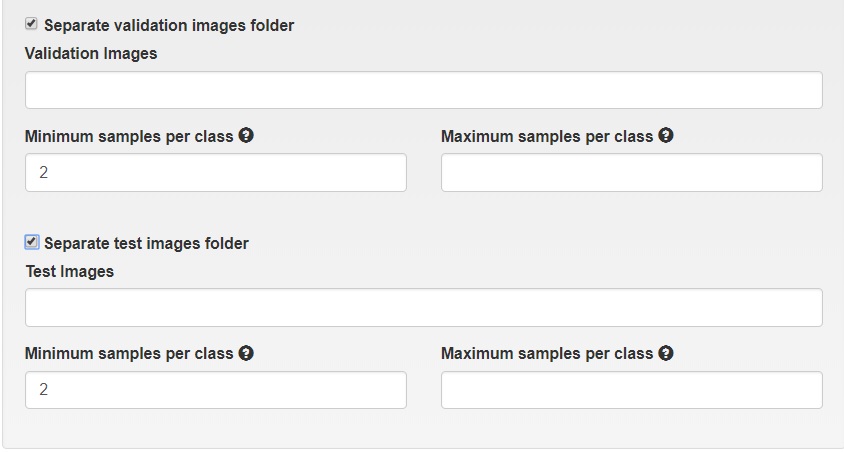
* Image Type – тип изображения (черно-белое или цветное)
* Image size (Width x Height) – Размер изображений (все изображения будут приведены к одномц размеру)
* Resize Transformation – опции для изменения соотношения сторон
* Training Images - папка, содержащая вложенные папки с изображениями.
* Minimum samples per class – минимальное количество изображений для каждого класса (можно оставить пустым – параметр будет проигнорирован)
* Maximum samples per class - максимальное количество изображений для каждого класса (можно оставить пустым – параметр будет проигнорирован)
* % for validation - процент изображений для валидации из обучающего набора
* % for testing – процент тестовых изображений из обучающего набора
* Separate validation images folder – отдельная папка с изображениями для валидации
* Separate test images folder – отдельная папка с тестовыми изображениями
* Image Encoding – тип изображений (все изоражения приводятся к определенному типу)
* Group Name – название группы
* Dataset Name – название набора

1. Для обучения нейронной сети потребуется DataSet. Для этого необходимо сформировать: трейновую, тестовую и валиационную часть.

* Изображения для обучения нейронной сети нужно распределить по папкам, название папки – желаемая метка класса (label): 0, 1 и 2. В каждой папке находятся изображения членов класса. Полученные папки вложить в одну папку – Train\_Data.

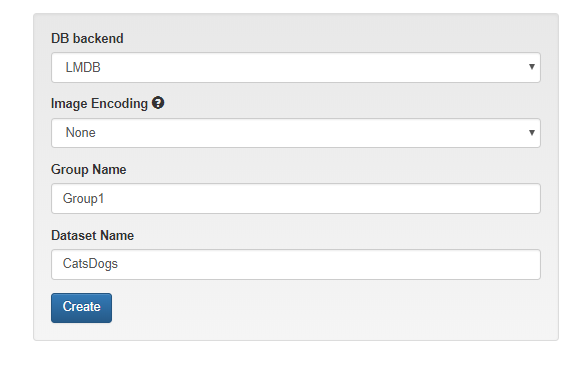


* При необходимости тестовые и валидационные изображения распределяются аналогичным образом в папки – Test\_Data, Val\_Data либо просто указывается процент от Train\_Data. Для этого не обходимо поставить галочки и описать минимальное и максимальное количество сэмплов в классе.

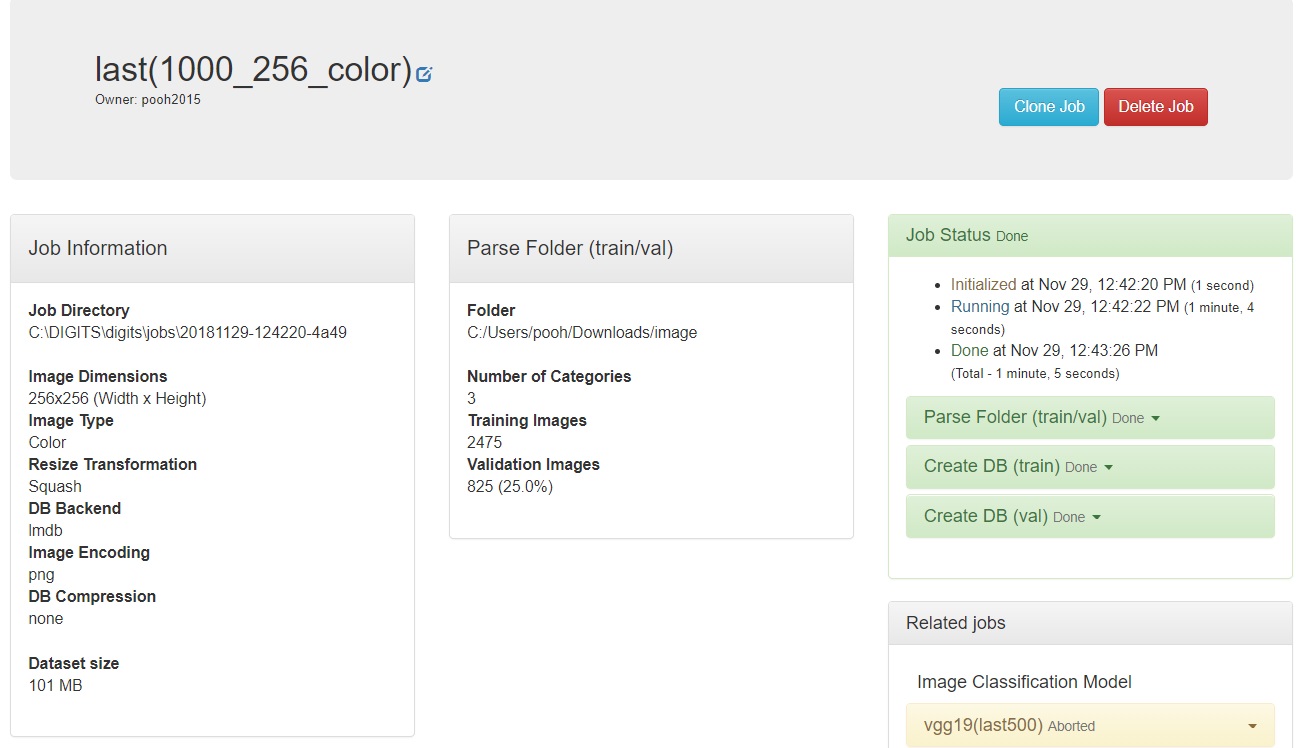


1. Далее необходимо заполнить поля в форме. Пример заполнения:



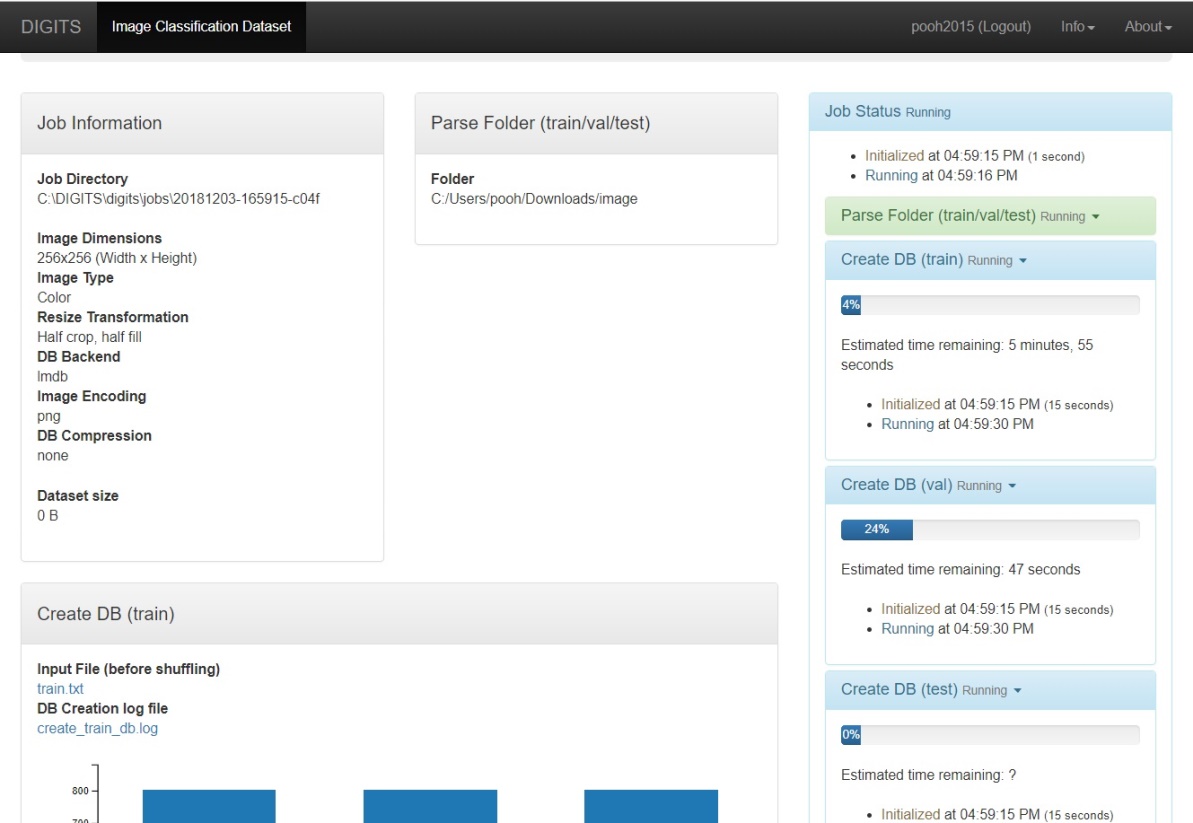


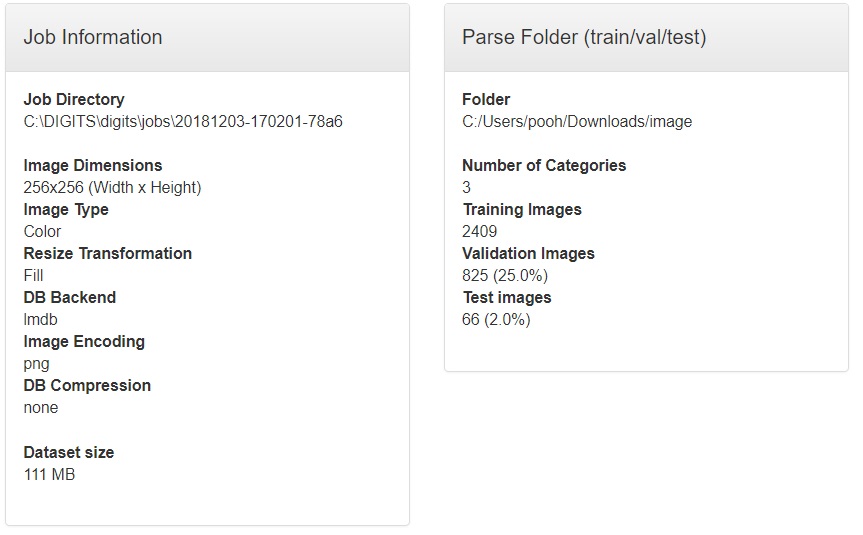
1. Нажать Create → Открывается форма с прогресс барами выполнения каждого из этапов → Об успешном завершении информирует положительный статус.

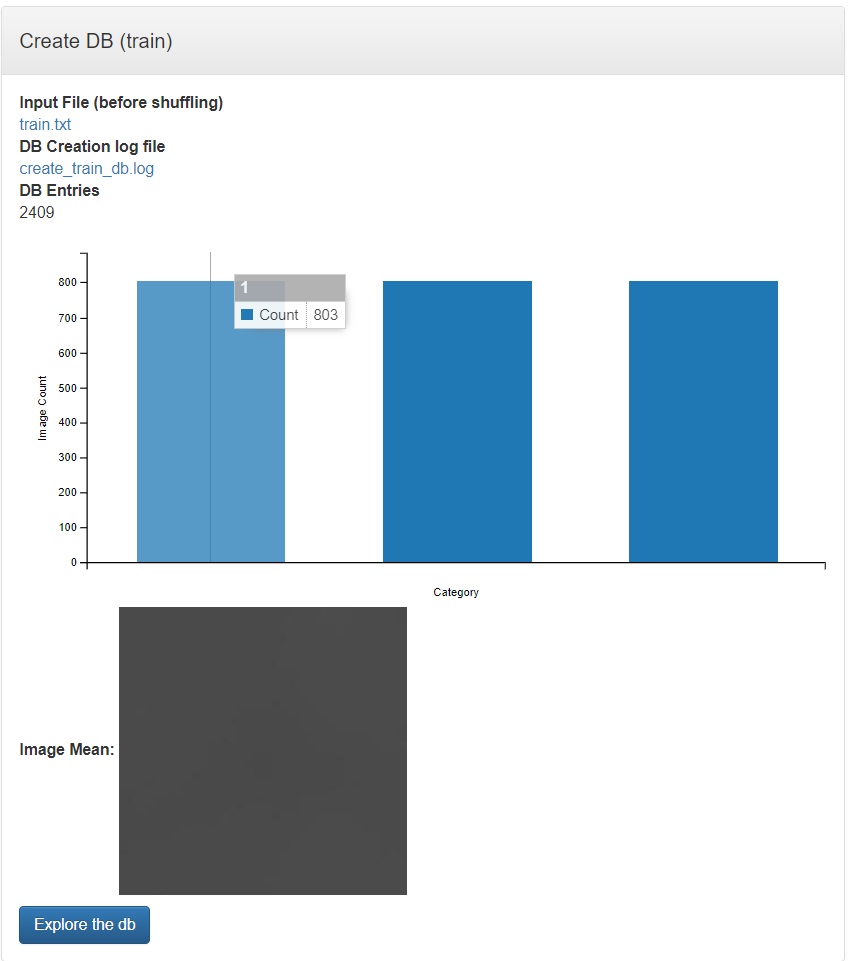


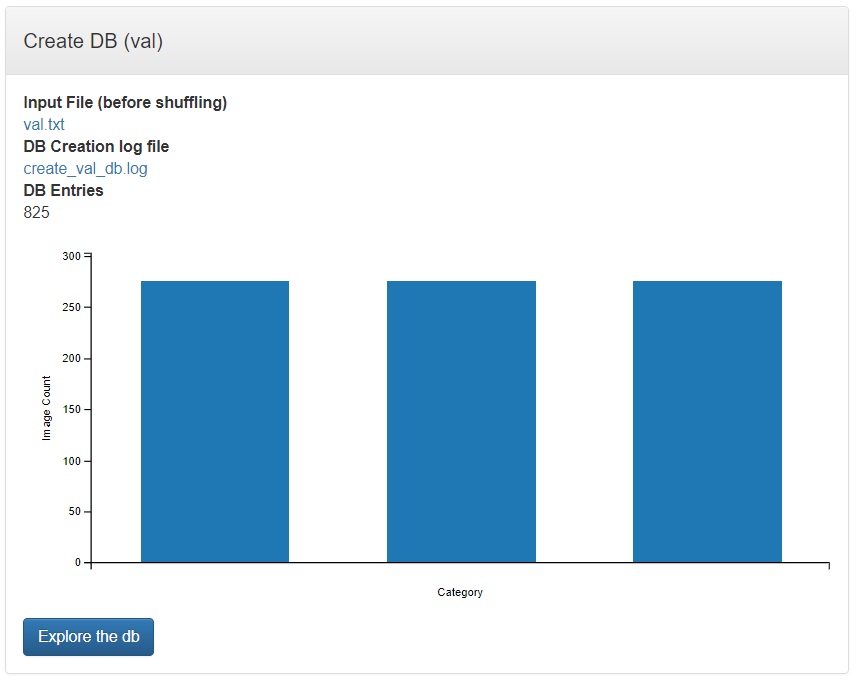
В результате можно увидеть:

* Время, затраченное на парсинг и создание данных.
* Число классов
* Число обучающих, валидационных и тестовых изображений
* Среднее изображение классов
* Изображения каждого из наборов

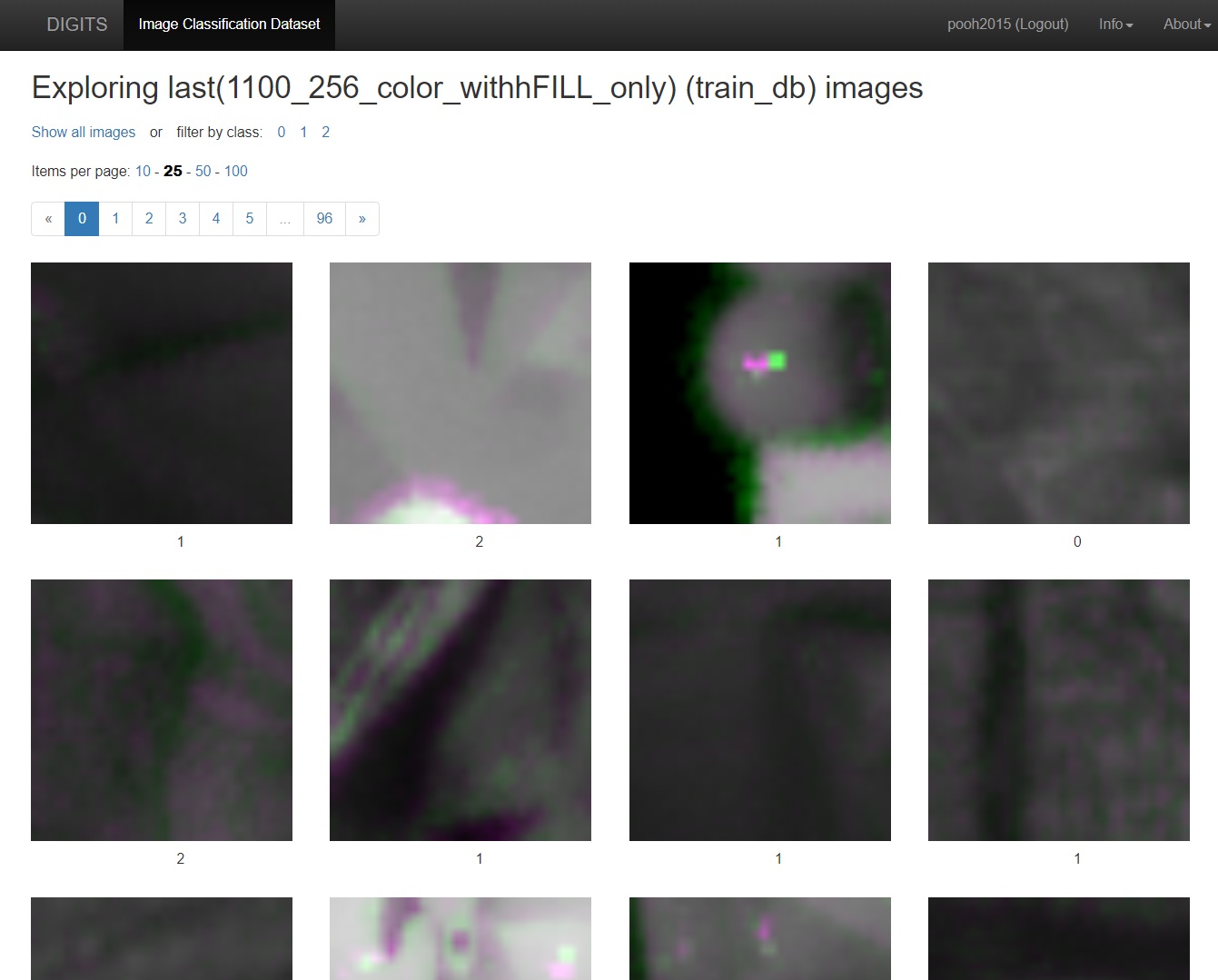




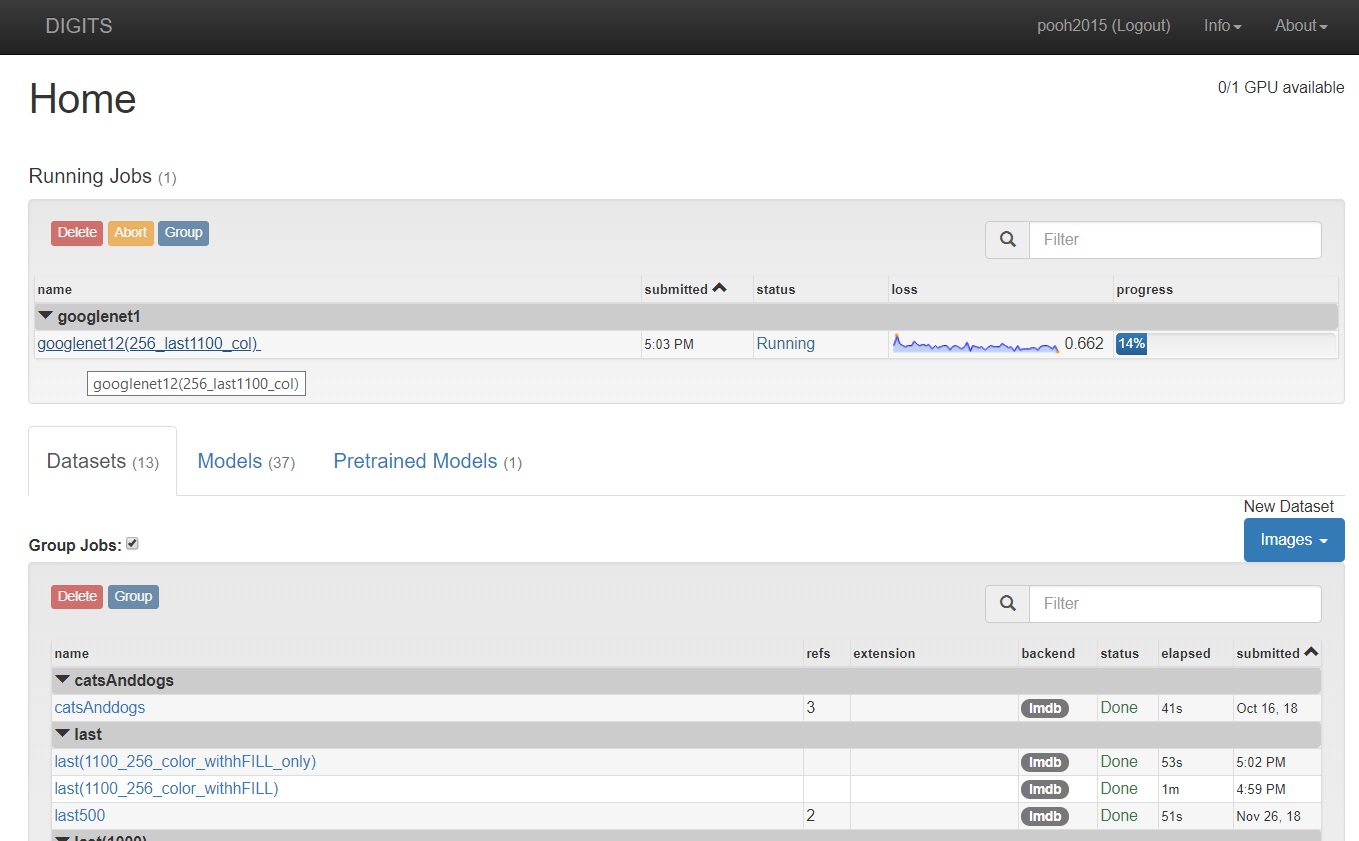






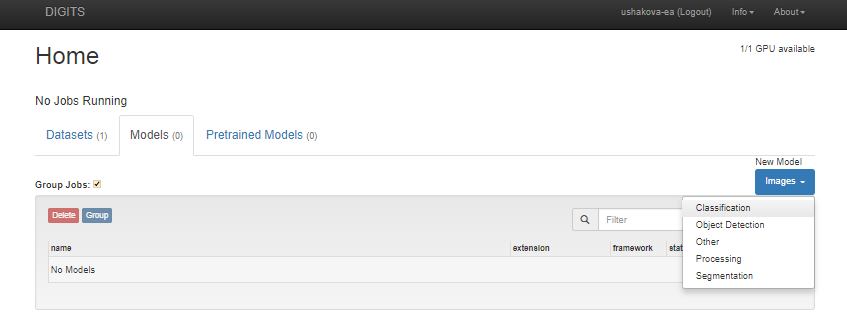


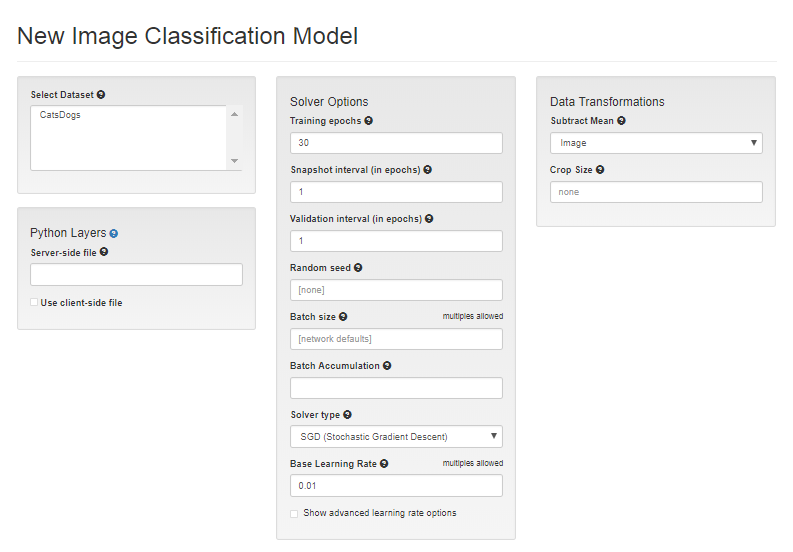
1. Сформированный DataSet будет сохранен во вкладке DataSets

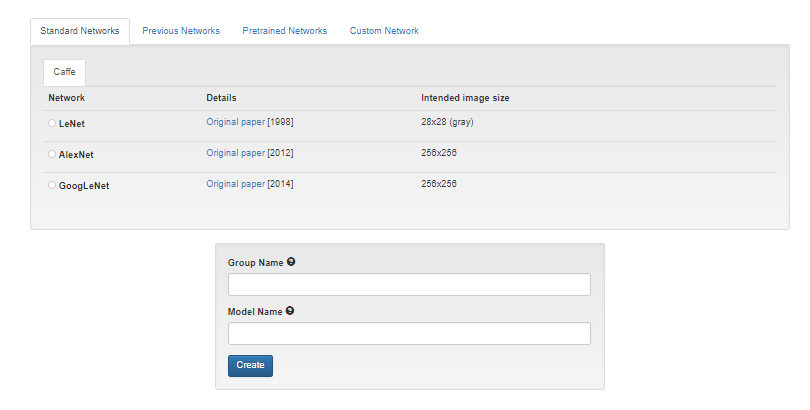


## Формирование модели

1. Для формирования модели для решения задачи классификации необходимо открыть вкладку Models и выбрать Classification.



1. Далее откроется форма



1. В форме присутствуют следующие параметры:

* Select Dataset - набор данных для модели

Слои Python:

* Server-side file – файл на сервере, содержащий определение слоев (можно использовать файл на стороне клиента)

Настройки для решения:

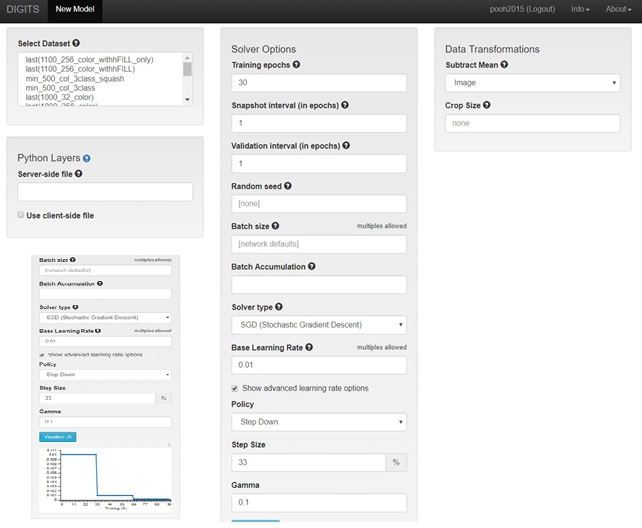
* Training epochs – количество эпох для обучения (циклов переобучения)
* Snapshot interval (in epochs) – количество эпох обучения между съемками
* Validation interval (in epochs) - количество эпох обучения между прохождением через один проход валидации
* Random seed - случайное начальное значение
* Batch size – размер батча (сколько изображений обрабатывается в один такт), зависит от характеристик видео карты. Тоесть чем больше памяти имеет видеокарта, тем большее количество сэмплов класса может обработать нейронная сеть за один такт. (Параметр подбирается индивидуально для каждого АРМ).
* Batch Accumulation – накопление градиентов по нескольким пакетам (полезно, когда вам нужен больший размер пакета для обучения, но он не помещается в память)
* Solver type – стратегия изменения весов нейронной сети
* Base Learning Rate – скорость обучения сети (принимает список через запятую)

Дополнительные параметры скорости обучения:

* Policy – правило
* Step Size – шаг в процентах от общего числа обучающей выборки.
* Gamma - параметр

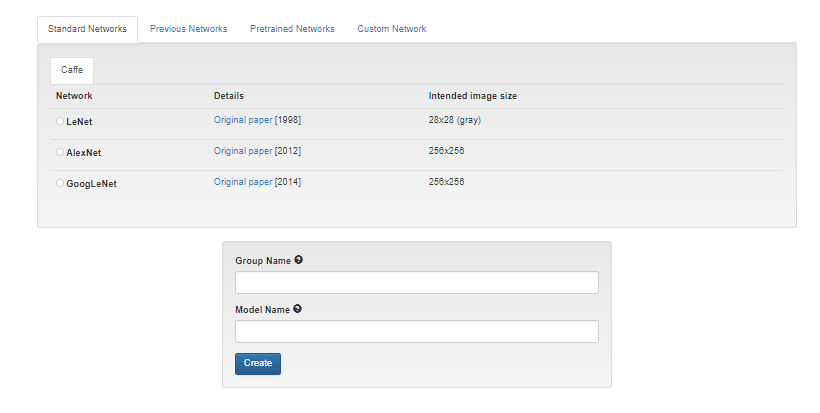
Преобазование данных:

* Subtract Mean – среднее изображение (вычитание среднего изображения из каждого изображения)
* Crop Size – размер кадра

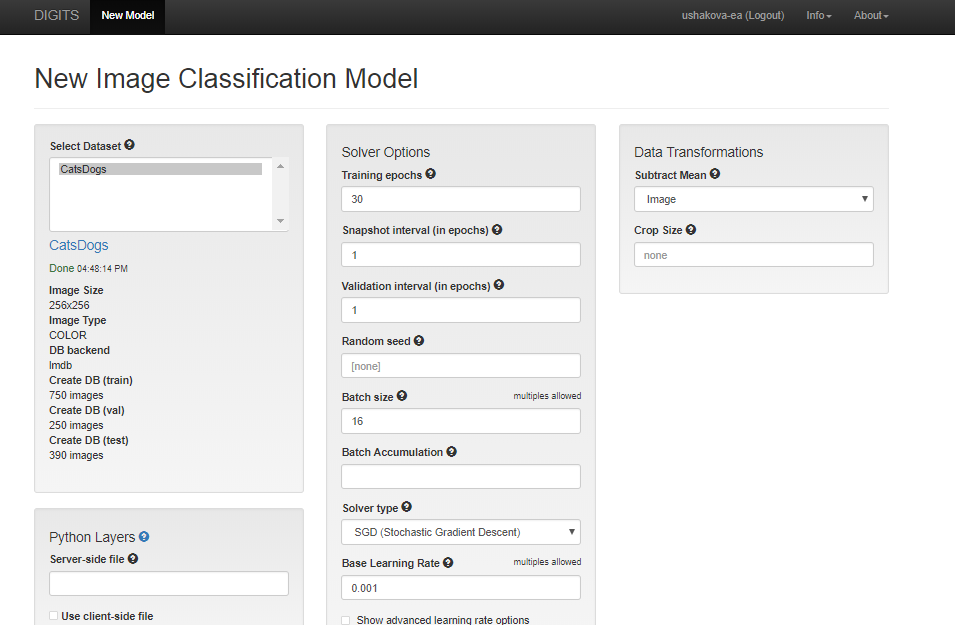


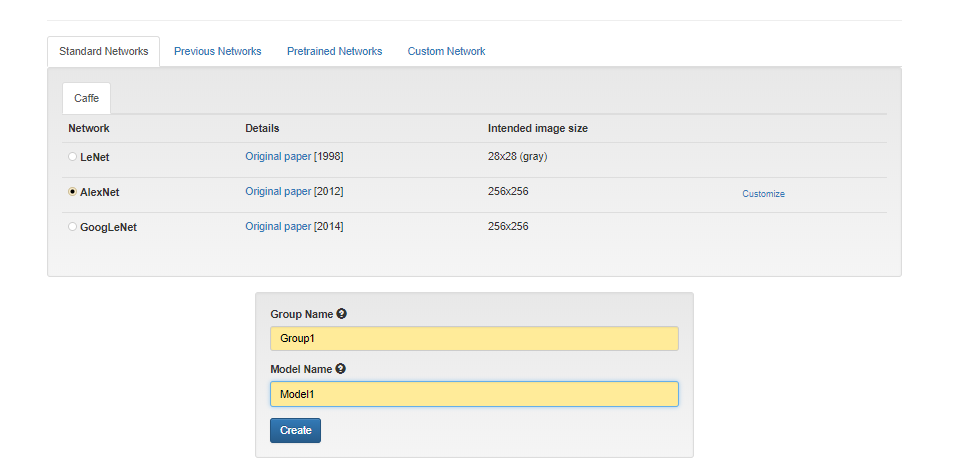
1. Для построения модели можно выбрать:

* Одну из трех стандартных сетей **LeNet**, **AlexNet, GoogLeNet**
* Ранее сформированную сеть
* Перетренированную сеть
* Задать свою сеть с нуля.

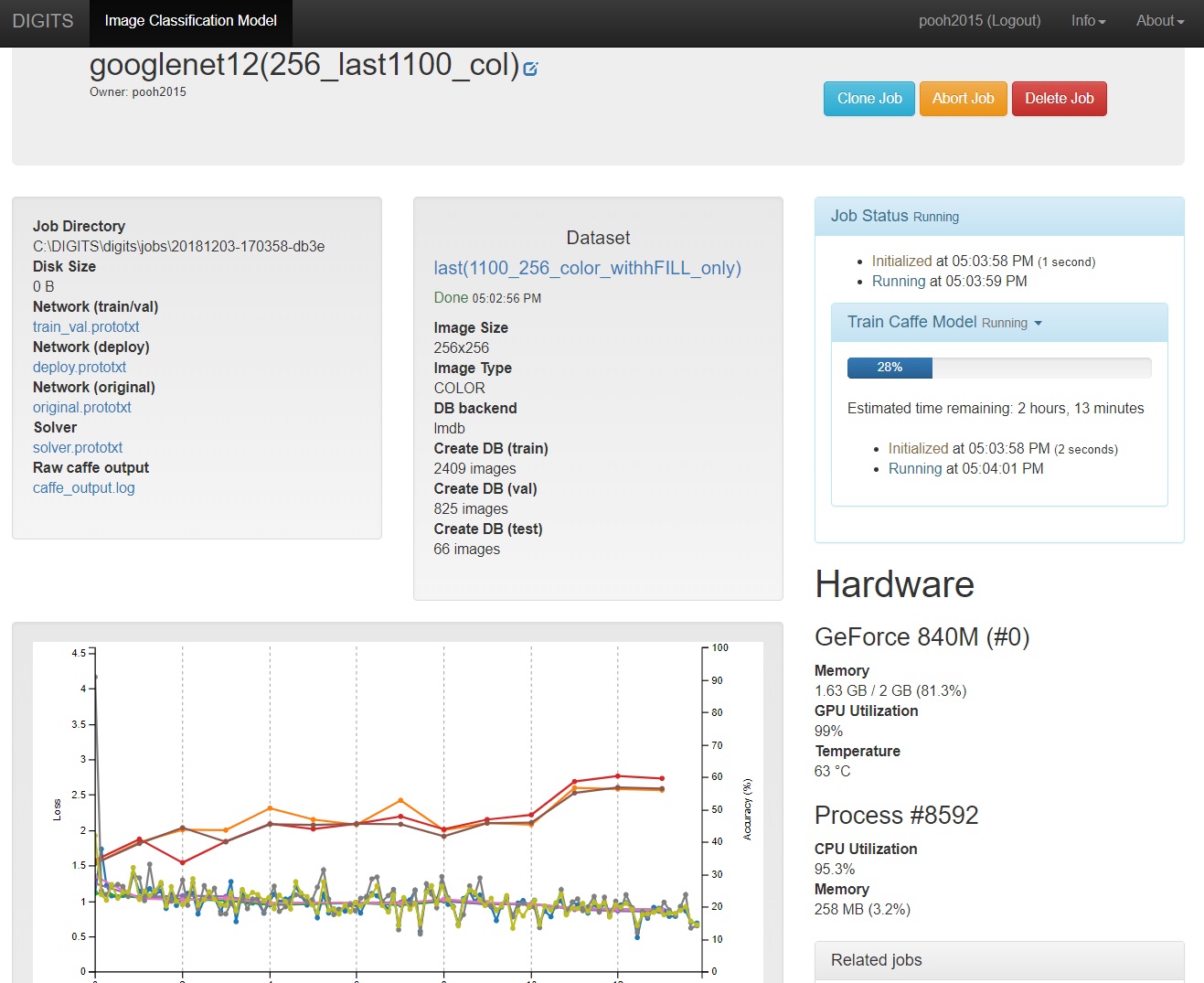


1. Далее необходимо заполнить поля в форме. Пример заполнения:





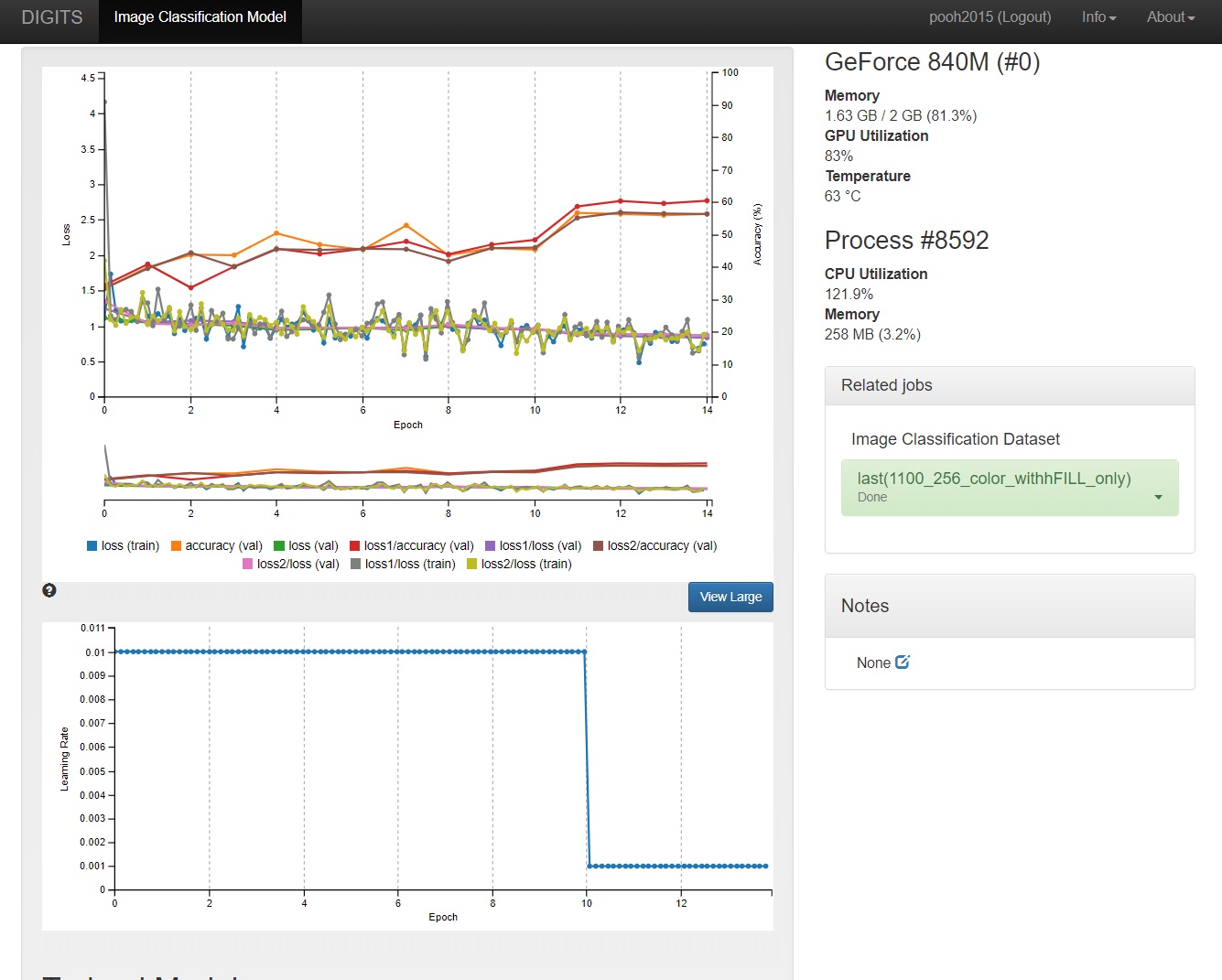
1. Нажать Create → Открывается форма с прогресс баром выполнения → Об успешном завершении информирует положительный статус - Done.



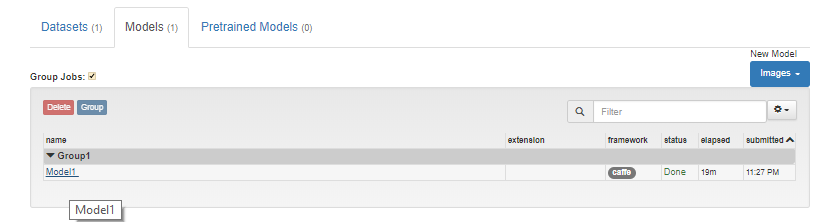
В процессе обучения, вы будите наблюдать справа: нагрузку на свои ресурсы, а именно на видеокарту и процессор.

В результате можно увидеть:

* Время, затраченное на формирование модели
* Количество использованной памяти на видеокарте и процессоре
* Ошибку на трейновом наборе
* Ошибку на тестовом наборе
* Точность валидации
* Скорость обучения



1. Модель будет сохранена во вкладке Models

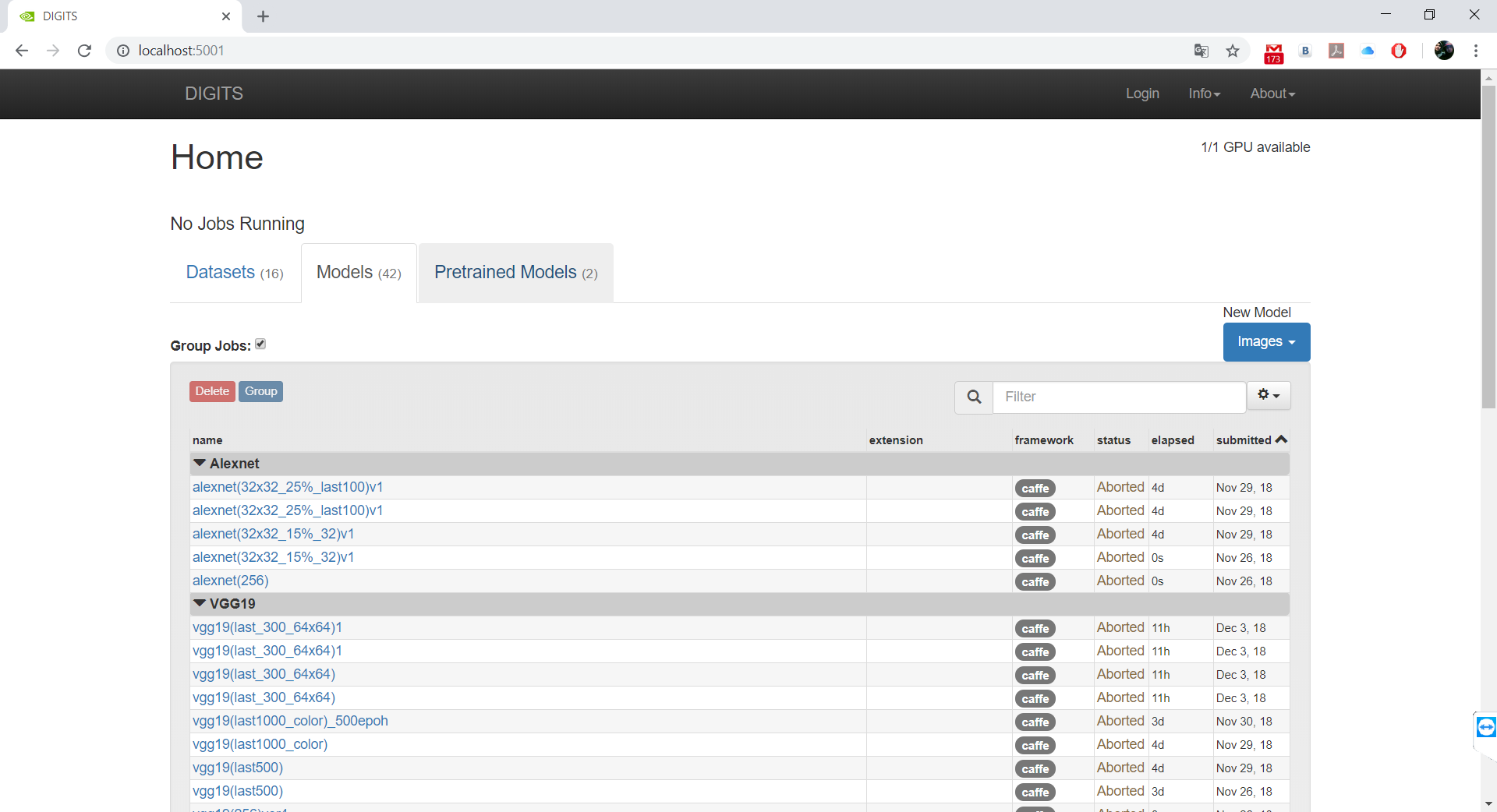


# Технология дообучения уже существующей модели нейронной сети

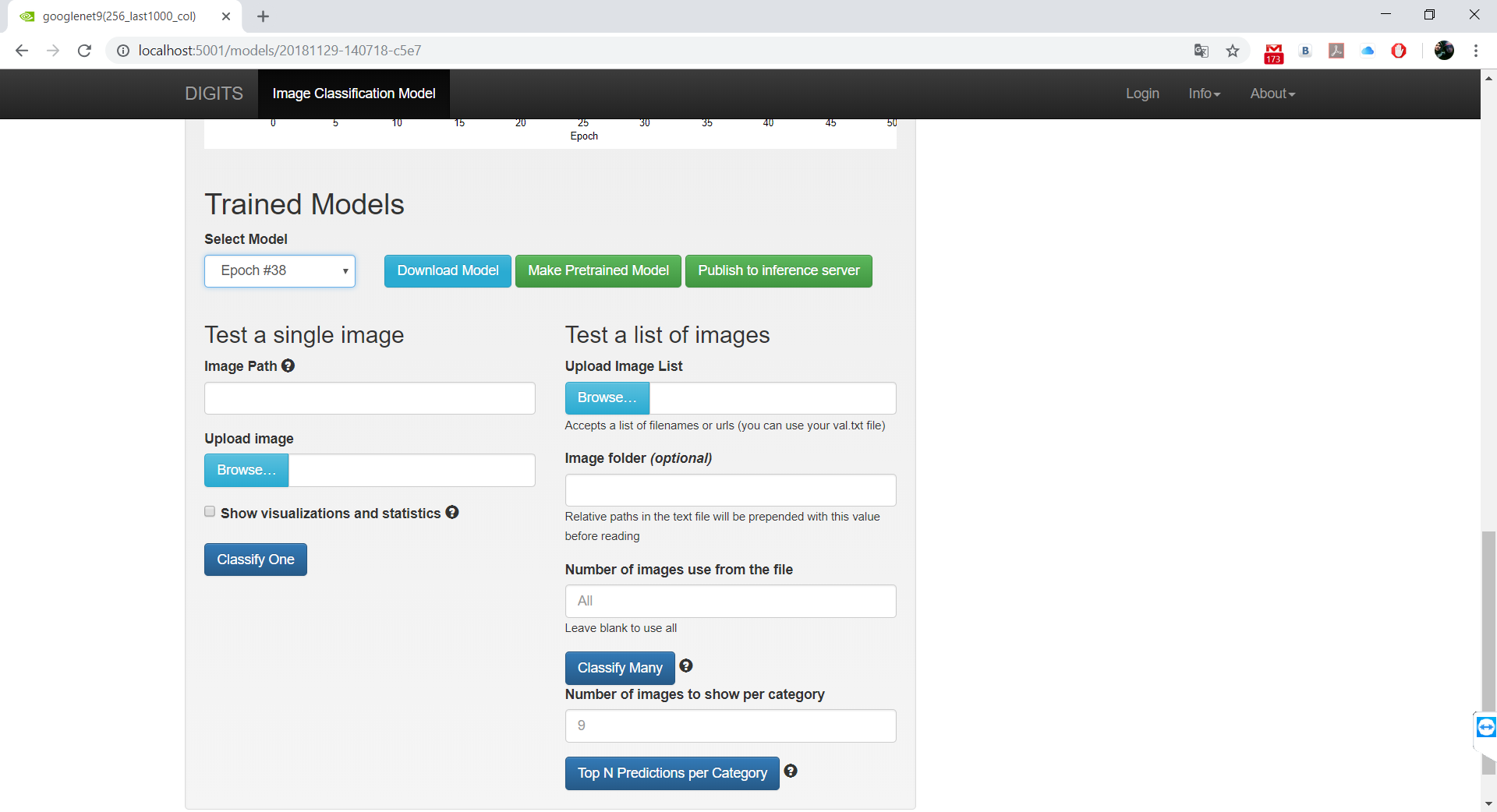
## 2. 1. Выгрузка модели

Для выгрузки модели нейронной сети, необходимо:

1. зайти в NVIDIA DIGITS во вкладку Models



1. Выбираем нужную модель и нажимаем Download Model (можно выбрать нужную эпоху из обученной модели, тем самым получить наилучший результат среди всех эпох за все время обучения)



1. По окончанию генерации нужной эпохи модели она сохраняется на локальную директорию. В итоге наша модель выглядит следующим образом



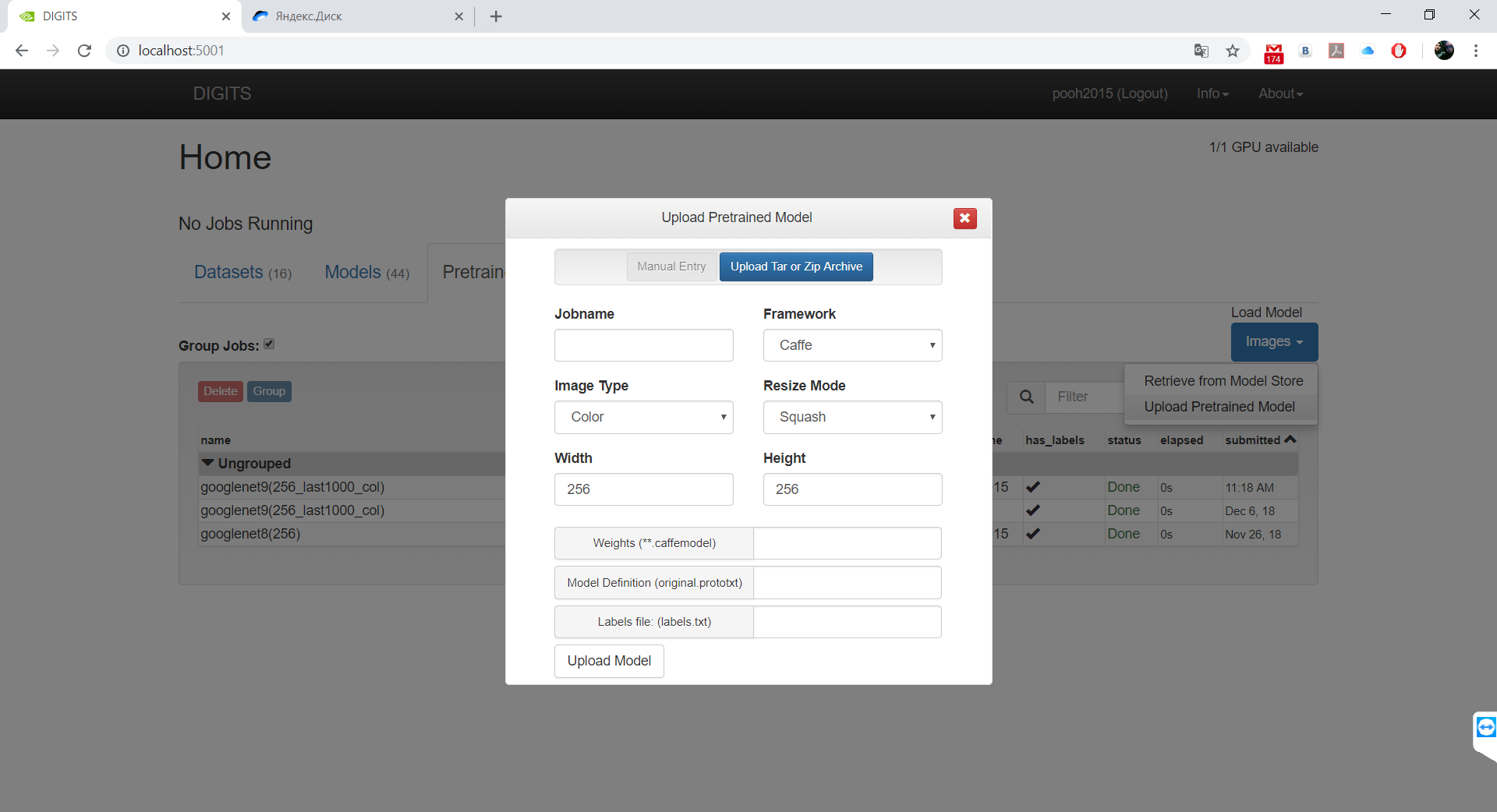
## 2. 2. Дообучение существующей модели

Для дообучения ранее скаченной (преобученной) модели необходимо:

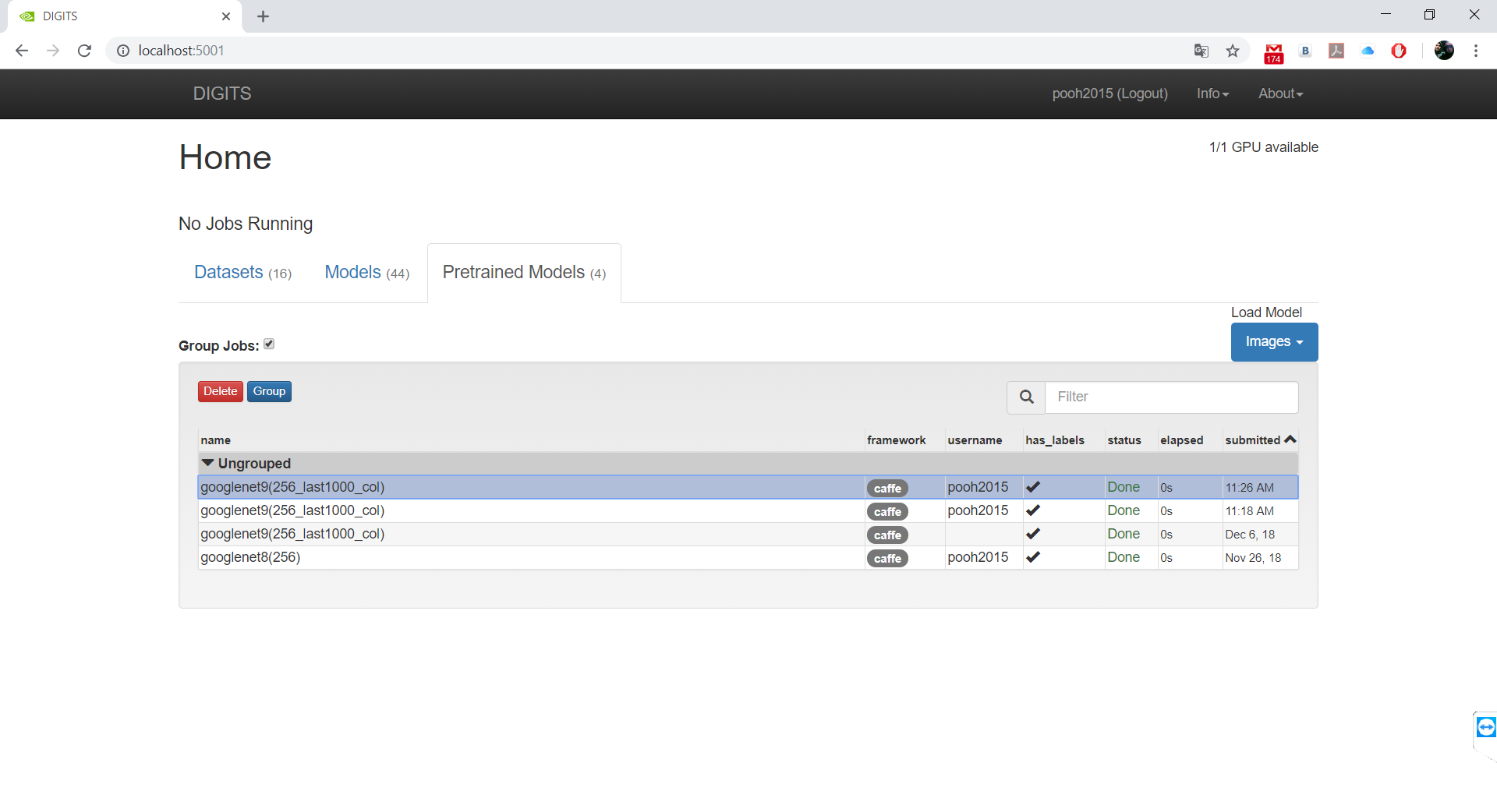
1. Зайти в NVIDIA DIGITS во вкладку Pretrained Models и выбираем в Load Model → Upload Pretrained Model



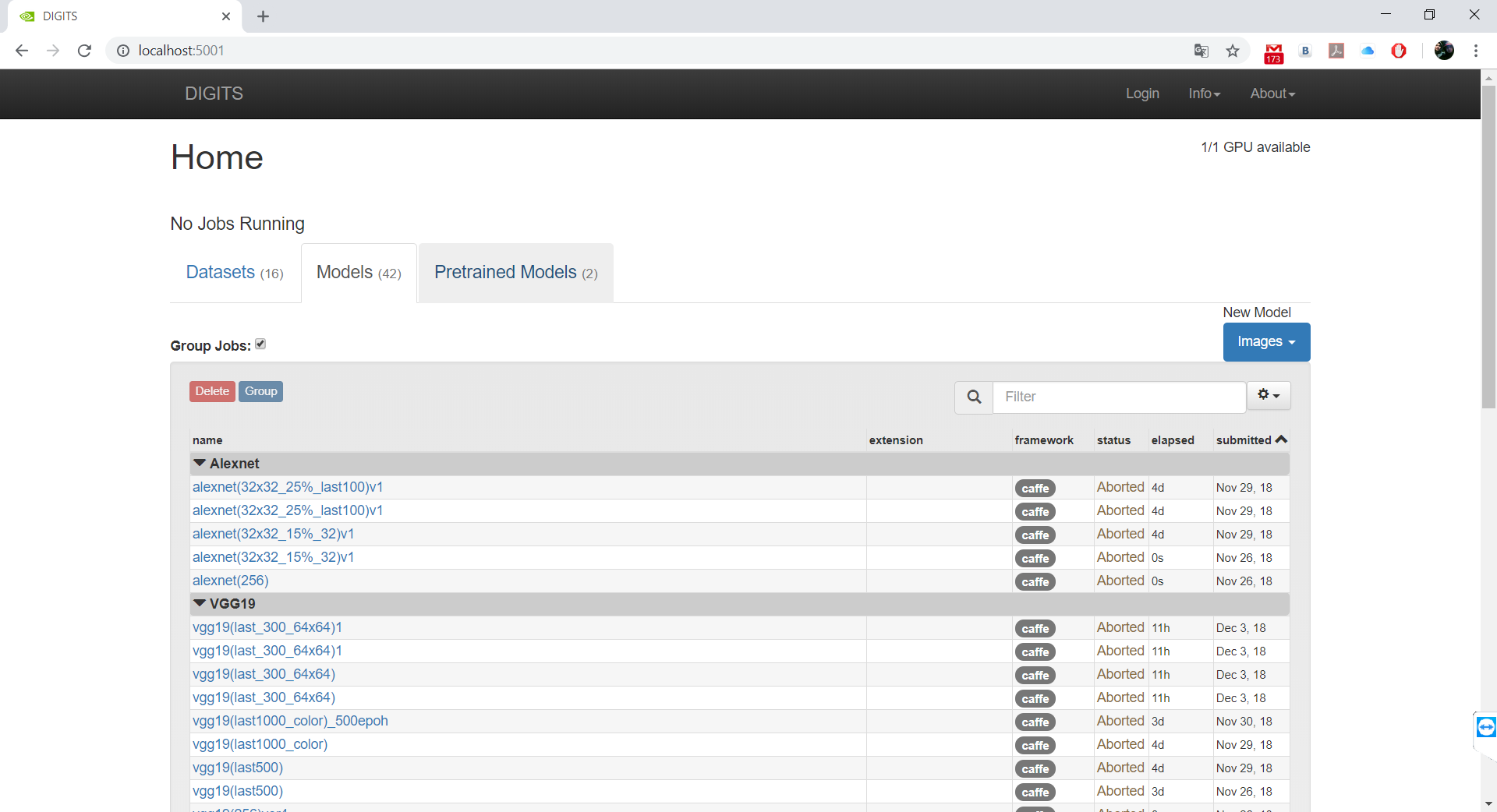
1. В открывшемся диалоговом окне выбираем Upload Tar or Zip Archive



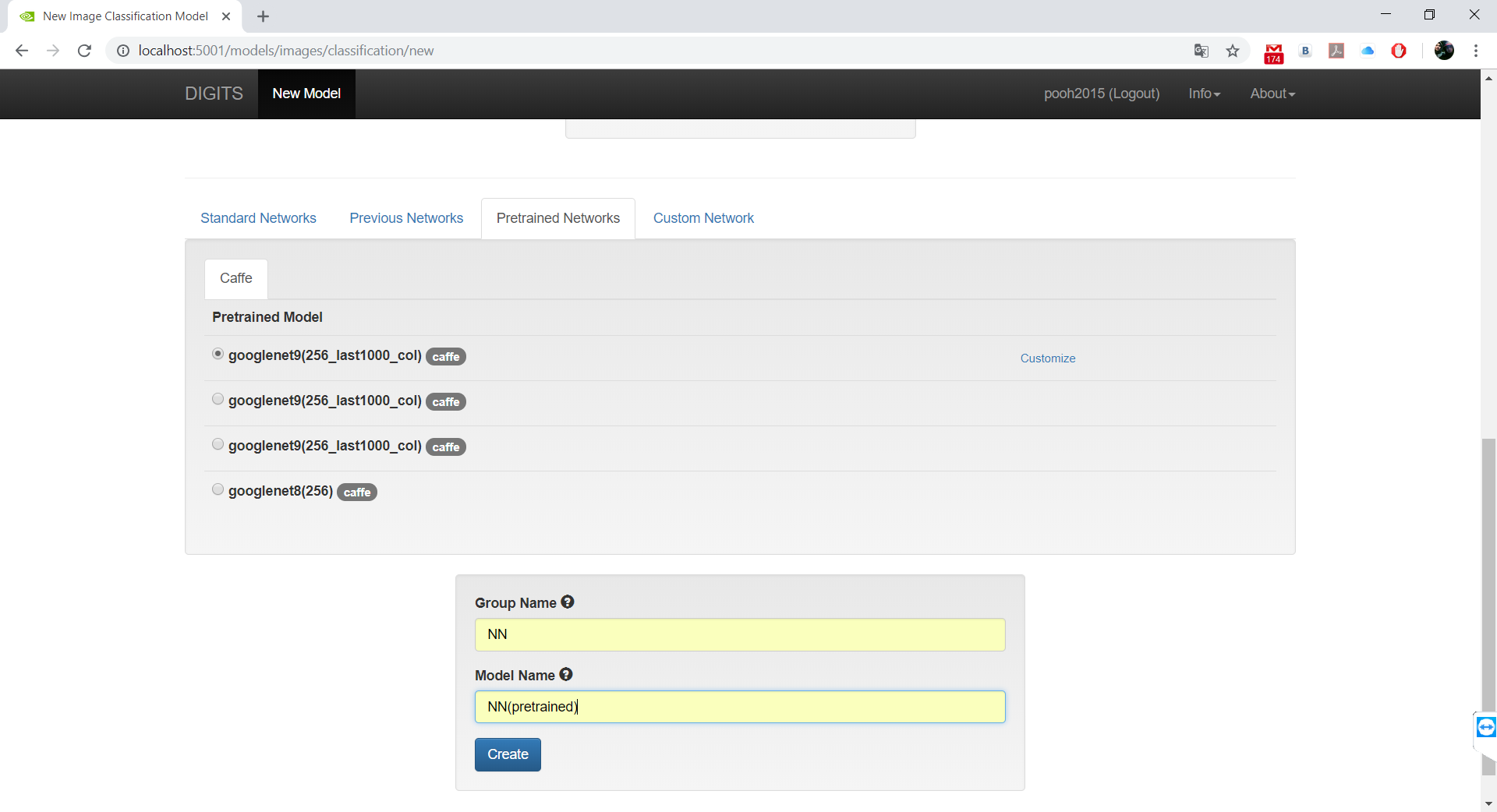
1. Выбираем скаченную претренированную модель и нажимаем Открыть и наше нейронная сеть подгружается в нашу систему DIGITS.



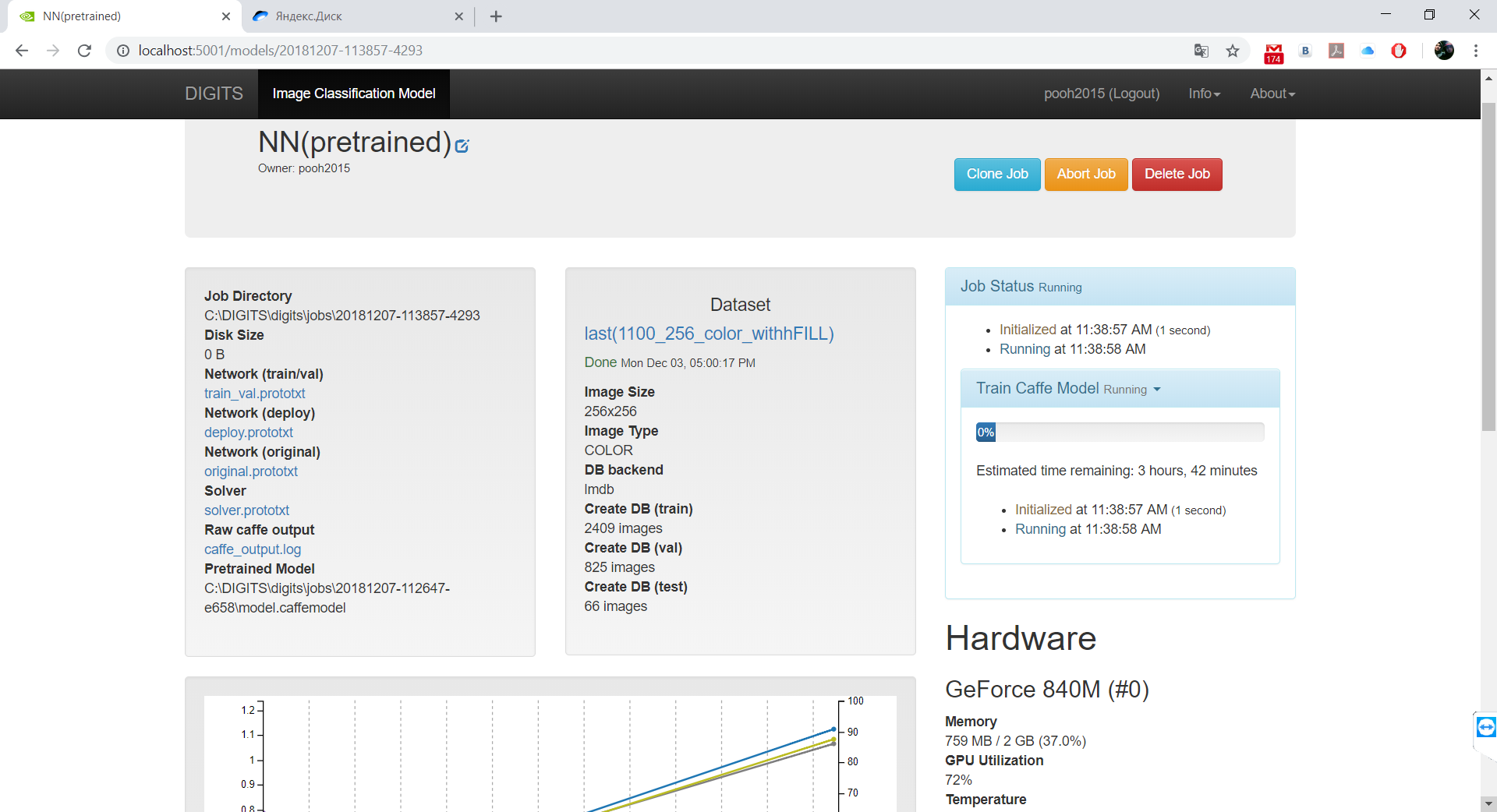
1. Зайти в NVIDIA DIGITS во вкладку Models и выбрать в New Model → Classification



1. Далее заходи во вкладку Pretrained Networks и выбираем нужную нам претренированную сеть. И одновременно выбираем нужный нам параметры обучения и dataset на котором собираемся до обучать.



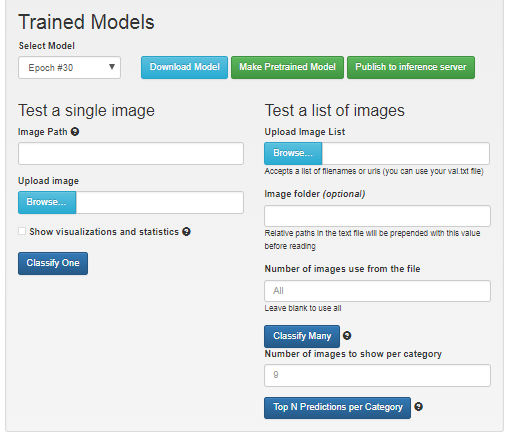
1. После ввода всех данных наживает Create и наша сеть начинает процесс дообучения.



# Тестирование модели

Существует возможность проверить:

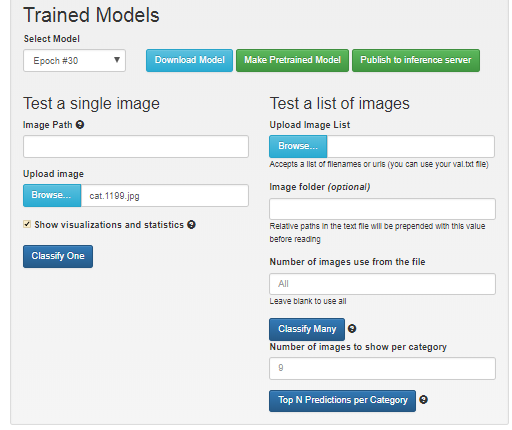
* Список изображений
* Одно изображение



Для тестирования одного изображения

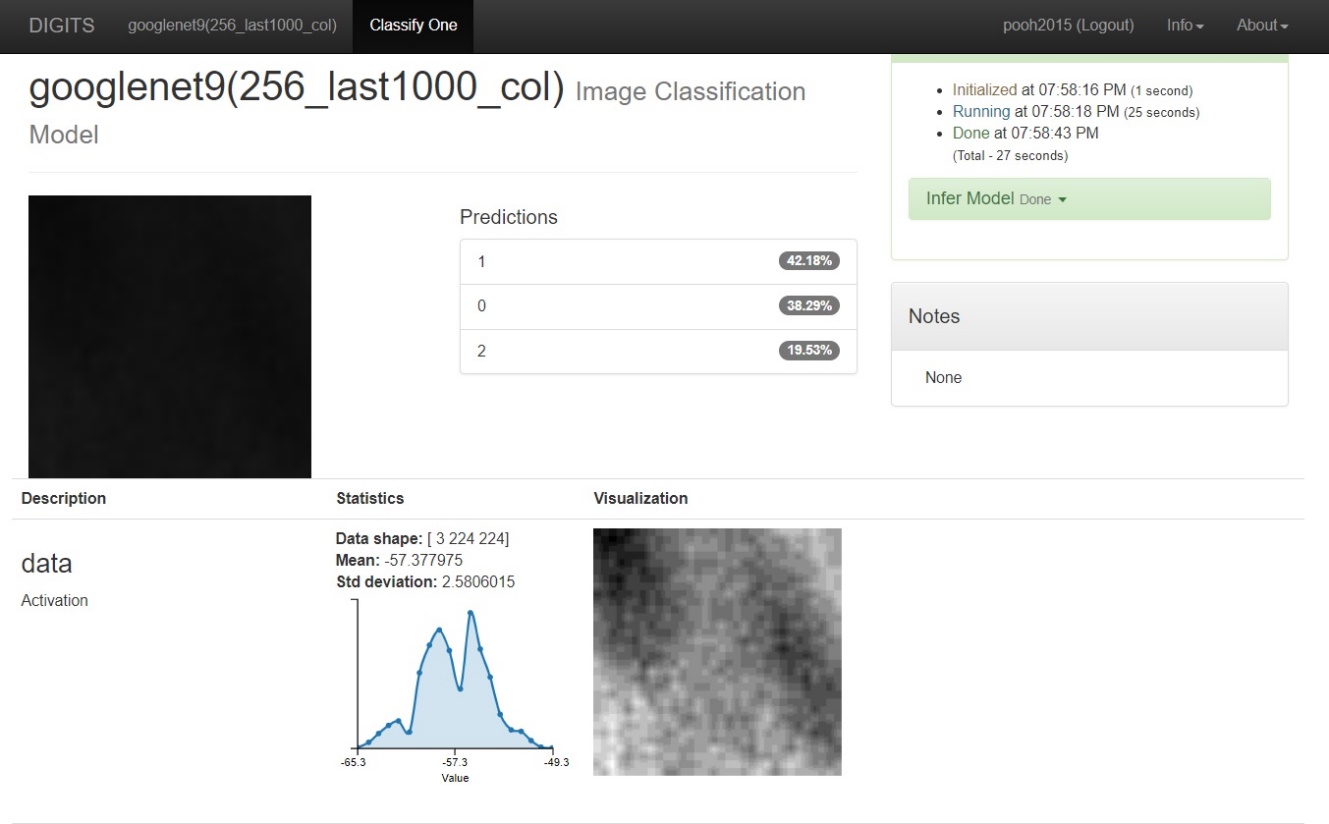
* Загрузить изображение
* Нажать Classify One

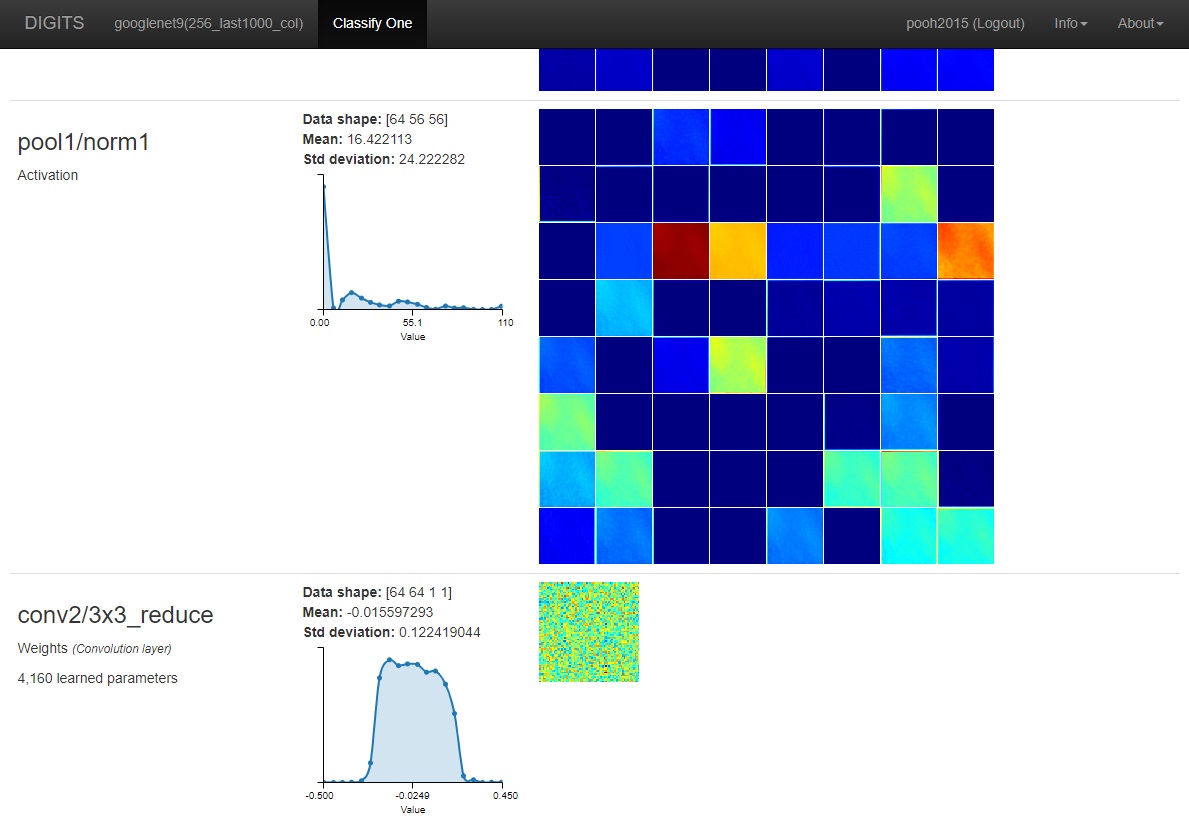
Например:

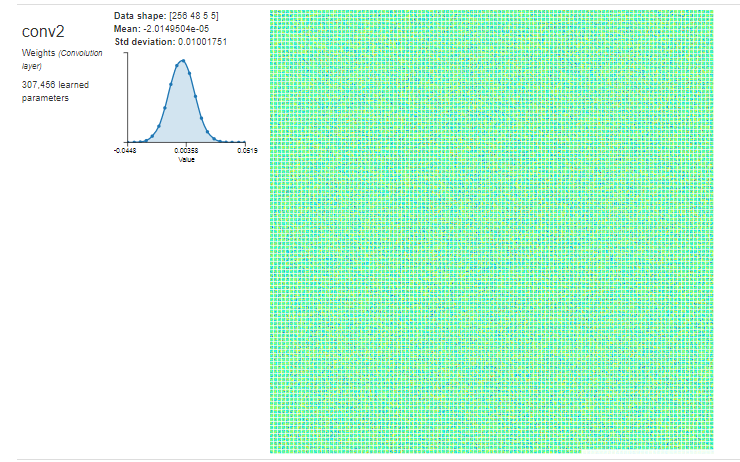


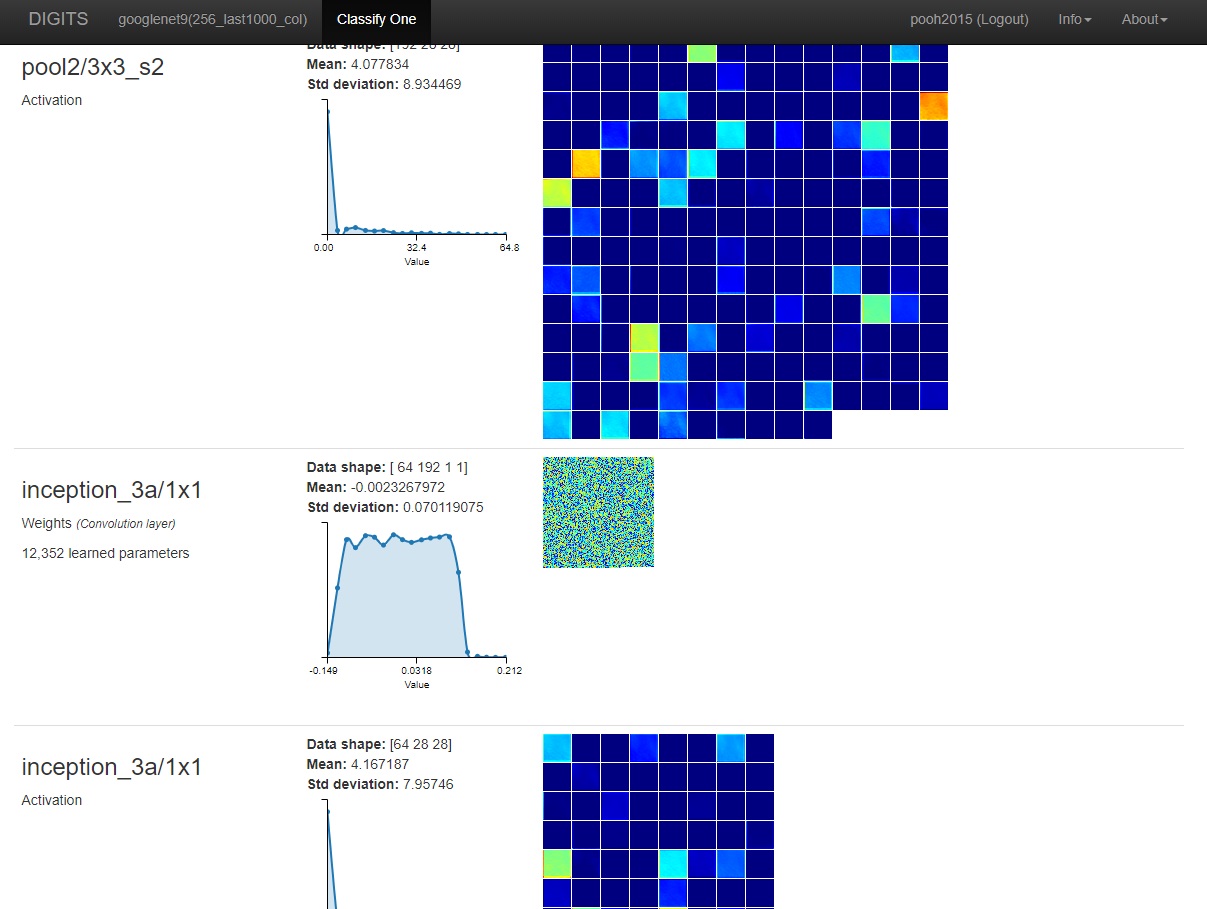
В результате можно увидеть:

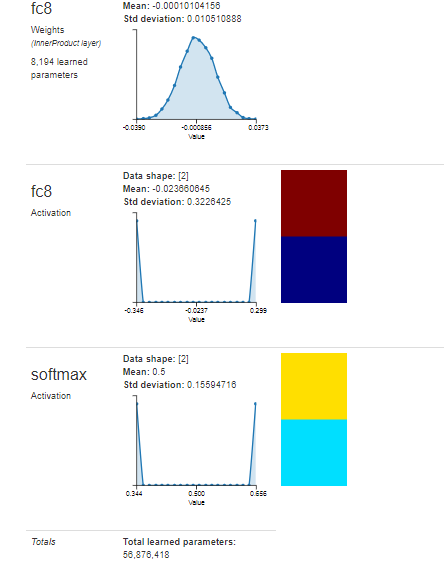
* Процент принадлежности к тому или иному классу
* Статистику и визуализацию по нормализации, пулингу и сверточным слоям
* График функции ошибки
* Количесто изученных параметров

****









# Список используемых источников

1. Техническое задание на на опытно-конструкторскую разработку «Построение и обучение нейронной сети для идентификации фокусного расстояния по серии изображений.» (ПО «SmartGetDistance»)». № 1 от 16.10.2018 г. Нижний Новгород, 2018.
2. Инструкция по развертыванию нейронной сети NVIDIA DIGITS **«**Построение и обучение нейронной сети для идентификации фокусного расстояния по серии изображений**» («**SmartGetDistance**»**), г. Нижний Новгород, 2018.