Artificial Intelligence Nanodegree

Computer Vision Capstone

Project: Facial Keypoint Detection

Welcome to the final Computer Vision project in the Artificial Intelligence Nanodegree program!

In this project, you'll combine your knowledge of computer vision techniques and deep learning to build and end-to-end facial keypoint recognition system! Facial keypoints include points around the eyes, nose, and mouth on any face and are used in many applications, from facial tracking to emotion recognition.

There are three main parts to this project:

Part 1 : Investigating OpenCV, pre-processing, and face detection

Part 2 : Training a Convolutional Neural Network (CNN) to detect facial keypoints

Part 3 : Putting parts 1 and 2 together to identify facial keypoints on any image!

**Here's what you need to know to complete the project:*

1. In this notebook, some template code has already been provided for you, and you will need to implement additional functionality to successfully complete this project. You will not need to modify the included code beyond what is requested.

   a. Sections that begin with **'(IMPLEMENTATION)'** in the header indicate that the following block of code will require additional functionality which you must provide. Instructions will be provided for each section, and the specifics of the implementation are marked in the code block with a 'TODO' statement. Please be sure to read the instructions carefully!

1. In addition to implementing code, there will be questions that you must answer which relate to the project and your implementation.

   a. Each section where you will answer a question is preceded by a **'Question X'** header.

   b. Carefully read each question and provide thorough answers in the following text boxes that begin with **'Answer:'**.

> **Note:** Code and Markdown cells can be executed using the **Shift + Enter** keyboard shortcut. Markdown cells can be edited by double-clicking the cell to enter edit mode.

The rubric contains **optional** suggestions for enhancing the project beyond the minimum requirements. If you decide to pursue the "(Optional)" sections, you should include the code in this IPython notebook.

Your project submission will be evaluated based on your answers to *each* of the questions and the code implementations you provide.

Steps to Complete the Project

Each part of the notebook is further broken down into separate steps. Feel free to use the links below to navigate the notebook.

In this project you will get to explore a few of the many computer vision algorithms built into the OpenCV library. This expansive computer vision library is now almost 20 years old (https://en.wikipedia.org/wiki/OpenCV#History) and still growing!

The project itself is broken down into three large parts, then even further into separate steps. Make sure to read through each step, and complete any sections that begin with **'(IMPLEMENTATION)'** in the header; these implementation sections may contain multiple TODOs that will be marked in code. For convenience, we provide links to each of these steps below.

Part 1 : Investigating OpenCV, pre-processing, and face detection

- Step 0: Detect Faces Using a Haar Cascade Classifier
- Step 1: Add Eye Detection
- Step 2: De-noise an Image for Better Face Detection
- Step 3: Blur an Image and Perform Edge Detection
- Step 4: Automatically Hide the Identity of an Individual

Part 2 : Training a Convolutional Neural Network (CNN) to detect facial keypoints

- Step 5: Create a CNN to Recognize Facial Keypoints
- Step 6: Compile and Train the Model
- Step 7: Visualize the Loss and Answer Questions

Part 3 : Putting parts 1 and 2 together to identify facial keypoints on any image!

- Step 8: Build a Robust Facial Keypoints Detector (Complete the CV Pipeline)

Step 0: Detect Faces Using a Haar Cascade Classifier

Have you ever wondered how Facebook automatically tags images with your friends' faces? Or how high-end cameras automatically find and focus on a certain person's face? Applications like these depend heavily on the machine learning task known as *face detection* - which is the task of automatically finding faces in images containing people.

At its root face detection is a classification problem - that is a problem of distinguishing between distinct classes of things. With face detection these distinct classes are 1) images of human faces and 2) everything else.

We use OpenCV's implementation of Haar feature-based cascade classifiers (http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html) to detect human faces in images. OpenCV provides many pre-trained face detectors, stored as XML files on github (https://github.com/opencv/opencv/tree/master/data/haarcascades). We have downloaded one of these detectors and stored it in the `detector_architectures` directory.

Import Resources

In the next python cell, we load in the required libraries for this section of the project.

In [256]:
```
# Import required libraries for this section

%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
import math
import cv2                      # OpenCV library for computer vision
from PIL import Image
import time
```

Next, we load in and display a test image for performing face detection.

Note: by default OpenCV assumes the ordering of our image's color channels are Blue, then Green, then Red. This is slightly out of order with most image types we'll use in these experiments, whose color channels are ordered Red, then Green, then Blue. In order to switch the Blue and Red channels of our test image around we will use OpenCV's `cvtColor` function, which you can read more about by checking out some of its documentation located here (http://docs.opencv.org/3.2.0/df/d9d/tutorial_py_colorspaces.html). This is a general utility function that can do other transformations too like converting a color image to grayscale, and transforming a standard color image to HSV color space.

In [257]:
```
# Load in color image for face detection
image = cv2.imread('images/test_image_1.jpg')

# Convert the image to RGB colorspace
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Plot our image using subplots to specify a size and title
fig = plt.figure(figsize = (8,8))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('Original Image')
ax1.imshow(image)
```

Out[257]: <matplotlib.image.AxesImage at 0x205f75d4390>

There are a lot of people - and faces - in this picture. 13 faces to be exact! In the next code cell, we demonstrate how to use a Haar Cascade classifier to detect all the faces in this test image.

This face detector uses information about patterns of intensity in an image to reliably detect faces under varying light conditions. So, to use this face detector, we'll first convert the image from color to grayscale.

Then, we load in the fully trained architecture of the face detector -- found in the file *haarcascade_frontalface_default.xml* - and use it on our image to find faces!

To learn more about the parameters of the detector see this post (https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters).

In [3]:
```
# Convert the RGB  image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

# Extract the pre-trained face detector from an xml file
face_cascade = cv2.CascadeClassifier('detector_architectures/haarcascade_front
alface_default.xml')

# Detect the faces in image
faces = face_cascade.detectMultiScale(gray, 4, 6)

# Print the number of faces detected in the image
print('Number of faces detected:', len(faces))

# Make a copy of the orginal image to draw face detections on
image_with_detections = np.copy(image)

# Get the bounding box for each detected face
for (x,y,w,h) in faces:
    # Add a red bounding box to the detections image
    cv2.rectangle(image_with_detections, (x,y), (x+w,y+h), (255,0,0), 3)


# Display the image with the detections
fig = plt.figure(figsize = (8,8))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('Image with Face Detections')
ax1.imshow(image_with_detections)
```

```
Number of faces detected: 13
```

```
Out[3]: <matplotlib.image.AxesImage at 0x7fca4c7b1198>
```

In the above code, `faces` is a numpy array of detected faces, where each row corresponds to a detected face. Each detected face is a 1D array with four entries that specifies the bounding box of the detected face. The first two entries in the array (extracted in the above code as `x` and `y`) specify the horizontal and vertical positions of the top left corner of the bounding box. The last two entries in the array (extracted here as `w` and `h`) specify the width and height of the box.

Step 1: Add Eye Detections

There are other pre-trained detectors available that use a Haar Cascade Classifier - including full human body detectors, license plate detectors, and more. A full list of the pre-trained architectures can be found here (https://github.com/opencv/opencv/tree/master/data/haarcascades).

To test your eye detector, we'll first read in a new test image with just a single face.

In [4]:
```
# Load in color image for face detection
image = cv2.imread('images/james.jpg')

# Convert the image to RGB colorspace
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Plot the RGB image
fig = plt.figure(figsize = (6,6))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('Original Image')
ax1.imshow(image)
```

Out[4]: <matplotlib.image.AxesImage at 0x7fca4c7574e0>

Notice that even though the image is a black and white image, we have read it in as a color image and so it will still need to be converted to grayscale in order to perform the most accurate face detection.

So, the next steps will be to convert this image to grayscale, then load OpenCV's face detector and run it with parameters that detect this face accurately.

In [5]:
```
# Convert the RGB  image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

# Extract the pre-trained face detector from an xml file
face_cascade = cv2.CascadeClassifier('detector_architectures/haarcascade_front
alface_default.xml')

# Detect the faces in image
faces = face_cascade.detectMultiScale(gray, 1.25, 6)

# Print the number of faces detected in the image
print('Number of faces detected:', len(faces))

# Make a copy of the orginal image to draw face detections on
image_with_detections = np.copy(image)

# Get the bounding box for each detected face
for (x,y,w,h) in faces:
    # Add a red bounding box to the detections image
    cv2.rectangle(image_with_detections, (x,y), (x+w,y+h), (255,0,0), 3)


# Display the image with the detections
fig = plt.figure(figsize = (6,6))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('Image with Face Detection')
ax1.imshow(image_with_detections)
```

```
Number of faces detected: 1
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x7fca4c7024e0>
```

(IMPLEMENTATION) Add an eye detector to the current face detection setup.

A Haar-cascade eye detector can be included in the same way that the face detector was and, in this first task, it will be your job to do just this.

To set up an eye detector, use the stored parameters of the eye cascade detector, called `haarcascade_eye.xml`, located in the `detector_architectures` subdirectory. In the next code cell, create your eye detector and store its detections.

A few notes before you get started:

First, make sure to give your loaded eye detector the variable name

`eye_cascade`

and give the list of eye regions you detect the variable name

`eyes`

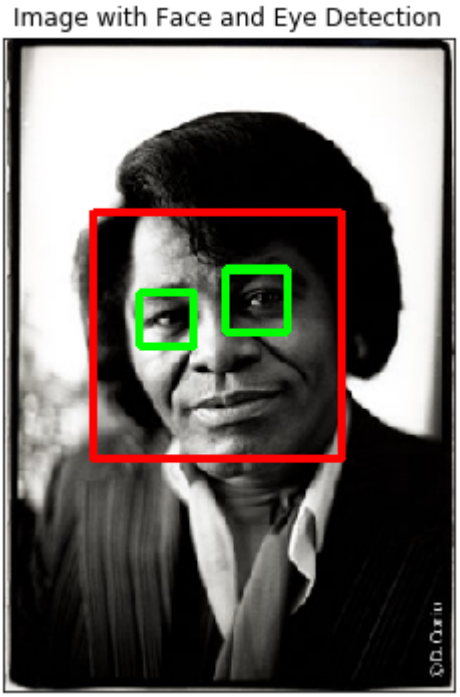
Second, since we've already run the face detector over this image, you should only search for eyes *within the rectangular face regions detected in* `faces`. This will minimize false detections.

Lastly, once you've run your eye detector over the facial detection region, you should display the RGB image with both the face detection boxes (in red) and your eye detections (in green) to verify that everything works as expected.

In [6]:
```
# Make a copy of the original image to plot rectangle detections
image_with_detections = np.copy(image)

# Loop over the detections and draw their corresponding face detection boxes
for (x,y,w,h) in faces:
    cv2.rectangle(image_with_detections, (x,y), (x+w,y+h),(255,0,0), 3)

# Do not change the code above this comment!


## TODO: Add eye detection, using haarcascade_eye.xml, to the current face det
ector algorithm

# Extract the pre-trained face detector from an xml file
eye_cascade= cv2.CascadeClassifier('detector_architectures/haarcascade_eye.xm
l')

eyes=[]
for (x,y,w,h) in faces:
    frame=gray[y:y+h,x:x+w]
    # Detect the faces in image
    eyes_detected = eye_cascade.detectMultiScale(frame)
    # realign the detection to the original image
    for (eye_x,eye_y,eye_w,eye_h) in eyes_detected:
        eyes.append((eye_x+x,eye_y+y,eye_w,eye_h))

# Print the number of faces detected in the image
print('Number of eyes detected:', len(eyes))


## TODO: Loop over the eye detections and draw their corresponding boxes in gr
een on image_with_detections

# Get the bounding box for each detected eye
for (x,y,w,h) in eyes:
    # Add a green bounding box to the detections image
    cv2.rectangle(image_with_detections, (x,y), (x+w,y+h), (0,255,0), 3)


# Plot the image with both faces and eyes detected
fig = plt.figure(figsize = (6,6))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('Image with Face and Eye Detection')
ax1.imshow(image_with_detections)
```

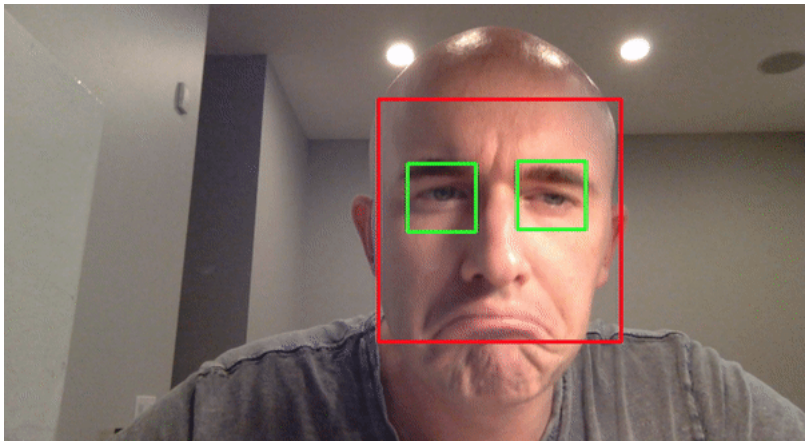
```
Number of eyes detected: 2
```

```
Out[6]: <matplotlib.image.AxesImage at 0x7fca4c6b4da0>
```



(Optional) Add face and eye detection to your laptop camera

It's time to kick it up a notch, and add face and eye detection to your laptop's camera! Afterwards, you'll be able to show off your creation like in the gif shown below - made with a completed version of the code!



Notice that not all of the detections here are perfect - and your result need not be perfect either. You should spend a small amount of time tuning the parameters of your detectors to get reasonable results, but don't hold out for perfection. If we wanted perfection we'd need to spend a ton of time tuning the parameters of each detector, cleaning up the input image frames, etc. You can think of this as more of a rapid prototype.

The next cell contains code for a wrapper function called `laptop_camera_face_eye_detector` that, when called, will activate your laptop's camera. You will place the relevant face and eye detection code in this wrapper function to implement face/eye detection and mark those detections on each image frame that your camera captures.

Before adding anything to the function, you can run it to get an idea of how it works - a small window should pop up showing you the live feed from your camera; you can press any key to close this window.

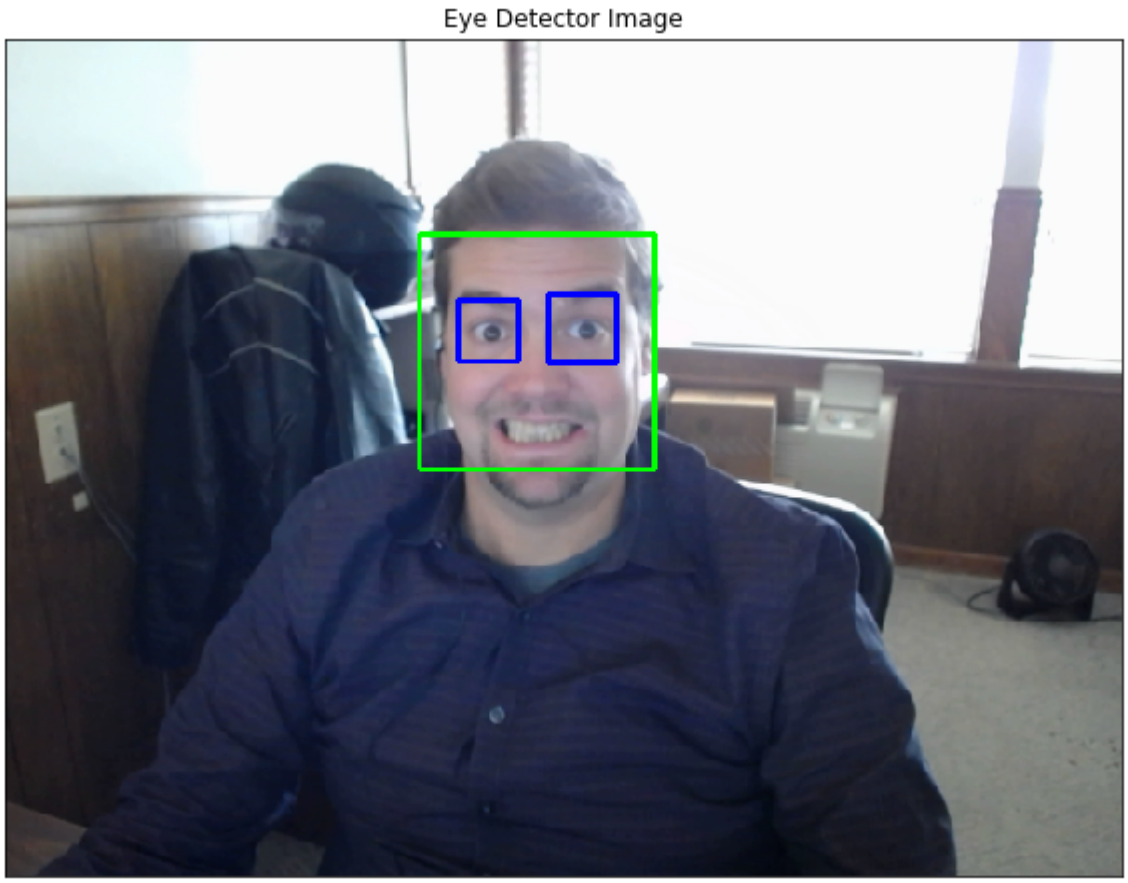
Note: Mac users may find that activating this function kills the kernel of their notebook every once in a while. If this happens to you, just restart your notebook's kernel, activate cell(s) containing any crucial import statements, and you'll be good to go!

In [396]:
```
def detect_face_eyes(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(
                                          gray,
                                          scaleFactor=1.1,
                                          minNeighbors=5,
                                          minSize=(30, 30),
                                          # flags=cv2.cv.CV_HAAR_SCALE_IMAGE
                                          )
    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

        eyes_detected = eye_cascade.detectMultiScale(gray[y:y+h,x:x+w])
        #realign the detection to the original image
        eyes=[]
        for (eye_x,eye_y,eye_w,eye_h) in eyes_detected:
            eyes.append((eye_x+x,eye_y+y,eye_w,eye_h))

        # Get the bounding box for each detected eye
        for (eye_x,eye_y,eye_w,eye_h) in eyes:
            # Add a green bounding box to the detections image
            cv2.rectangle(frame, (eye_x,eye_y), (eye_x+eye_w,eye_y+eye_h), (25
5,0,0), 2)
    return frame
```

In [398]:
```
### Add face and eye detection to this laptop camera function
# Make sure to draw out all faces/eyes found in each frame on the shown video
 feed

import cv2
import time

# wrapper function for face/eye detection with your laptop camera
def laptop_camera_go():
    # Create instance of video capturer
    cv2.namedWindow("face detection activated")
    vc = cv2.VideoCapture(1)

    # Try to get the first frame
    if vc.isOpened():
        rval, frame = vc.read()
    else:
        rval = False

    # Keep the video stream open
    while rval:
        # Plot the image from camera with all the face and eye detections mark
ed
        show_img=detect_face_eyes(frame)
        cv2.imshow("face detection activated", show_img)

        # Exit functionality - press any key to exit laptop video
        key = cv2.waitKey(20)
        if key > 0: # Exit by pressing any key
            # Destroy windows
            cv2.destroyAllWindows()

            # Make sure window closes on OSx
            for i in range (1,5):
                cv2.waitKey(1)
            return show_img

        # Read next frame
        time.sleep(0.05)             # control framerate for computation - def
ault 20 frames per sec
        rval, frame = vc.read()
```

In [405]:
```
# Call the laptop camera face/eye detector function above
last_frame=laptop_camera_go()

# Display the image
fig = plt.figure(figsize = (10,10))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.set_title('Eye Detector Image')
ax1.imshow( cv2.cvtColor(last_frame, cv2.COLOR_BGR2RGB))
```

Out[405]: <matplotlib.image.AxesImage at 0x205fef24518>



Step 2: De-noise an Image for Better Face Detection

Image quality is an important aspect of any computer vision task. Typically, when creating a set of images to train a deep learning network, significant care is taken to ensure that training images are free of visual noise or artifacts that hinder object detection. While computer vision algorithms - like a face detector - are typically trained on 'nice' data such as this, new test data doesn't always look so nice!

When applying a trained computer vision algorithm to a new piece of test data one often cleans it up first before feeding it in. This sort of cleaning - referred to as *pre-processing* - can include a number of cleaning phases like blurring, de-noising, color transformations, etc., and many of these tasks can be accomplished using OpenCV.

In this short subsection we explore OpenCV's noise-removal functionality to see how we can clean up a noisy image, which we then feed into our trained face detector.

Create a noisy image to work with

In the next cell, we create an artificial noisy version of the previous multi-face image. This is a little exaggerated - we don't typically get images that are this noisy - but image noise (https://digital-photography-school.com/how-to-avoid-and-reduce-noise-in-your-images/), or 'grainy-ness' in a digitial image - is a fairly common phenomenon.

In [9]:
```
# Load in the multi-face test image again
image = cv2.imread('images/test_image_1.jpg')

# Convert the image copy to RGB colorspace
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Make an array copy of this image
image_with_noise = np.asarray(image)

# Create noise - here we add noise sampled randomly from a Gaussian distributi
on: a common model for noise
noise_level = 40
noise = np.random.randn(image.shape[0],image.shape[1],image.shape[2])*noise_le
vel

# Add this noise to the array image copy
image_with_noise = image_with_noise + noise

# Convert back to uint8 format
image_with_noise = np.asarray([np.uint8(np.clip(i,0,255)) for i in image_with_
noise])

# Plot our noisy image!
fig = plt.figure(figsize = (8,8))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('Noisy Image')
ax1.imshow(image_with_noise)
```

```
Out[9]: <matplotlib.image.AxesImage at 0x7fd37403fd30>
```

In the context of face detection, the problem with an image like this is that - due to noise - we may miss some faces or get false detections.

In the next cell we apply the same trained OpenCV detector with the same settings as before, to see what sort of detections we get.

In [10]:
```
# Convert the RGB  image to grayscale
gray_noise = cv2.cvtColor(image_with_noise, cv2.COLOR_RGB2GRAY)

# Extract the pre-trained face detector from an xml file
face_cascade = cv2.CascadeClassifier('detector_architectures/haarcascade_front
alface_default.xml')

# Detect the faces in image
faces = face_cascade.detectMultiScale(gray_noise, 4, 6)

# Print the number of faces detected in the image
print('Number of faces detected:', len(faces))

# Make a copy of the orginal image to draw face detections on
image_with_detections = np.copy(image_with_noise)

# Get the bounding box for each detected face
for (x,y,w,h) in faces:
    # Add a red bounding box to the detections image
    cv2.rectangle(image_with_detections, (x,y), (x+w,y+h), (255,0,0), 3)


# Display the image with the detections
fig = plt.figure(figsize = (8,8))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('Noisy Image with Face Detections')
ax1.imshow(image_with_detections)
```

```
Number of faces detected: 13
```

```
Out[10]: <matplotlib.image.AxesImage at 0x7fd32e6ad4a8>
```

With this added noise we now miss one of the faces!

(IMPLEMENTATION) De-noise this image for better face detection

Time to get your hands dirty: using OpenCV's built in color image de-noising functionality called `fastNlMeansDenoisingColored` - de-noise this image enough so that all the faces in the image are properly detected. Once you have cleaned the image in the next cell, use the cell that follows to run our trained face detector over the cleaned image to check out its detections.

You can find its official documentation here ([documentation for denoising] (http://docs.opencv.org/trunk/d1/d79/group__photo__denoise.html#ga21abc1c8b0e15f78cd3eff672cb6c476) and a useful example here (http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_photo/py_non_local_means/py_non_local_means.html).

Note: you can keep all parameters *except* `photo_render` fixed as shown in the second link above. Play around with the value of this parameter - see how it affects the resulting cleaned image.

In [11]:
```
## TODO: Use OpenCV's built in color image de-noising function to clean up our
 noisy image!


denoised_image = cv2.fastNlMeansDenoisingColored(image_with_noise,None,20,10,7
,21)# your final de-noised image (should be RGB)

# Display the image de-noised
fig = plt.figure(figsize = (8,8))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('UN denoised_imageNoisy Image')
ax1.imshow(denoised_image)
```

Out[11]: <matplotlib.image.AxesImage at 0x7fd32e652128>

In [12]:
```
## TODO: Run the face detector on the de-noised image to improve your detectio
ns and display the result
# Convert the RGB  image to grayscale
gray_denoise = cv2.cvtColor(denoised_image, cv2.COLOR_RGB2GRAY)

# Extract the pre-trained face detector from an xml file
face_cascade = cv2.CascadeClassifier('detector_architectures/haarcascade_front
alface_default.xml')

# Detect the faces in image
faces = face_cascade.detectMultiScale(gray_denoise, 4, 6)

# Print the number of faces detected in the image
print('Number of faces detected:', len(faces))

# Make a copy of the orginal image to draw face detections on
image_with_detections = np.copy(image_with_noise)

# Get the bounding box for each detected face
for (x,y,w,h) in faces:
    # Add a red bounding box to the detections image
    cv2.rectangle(image_with_detections, (x,y), (x+w,y+h), (255,0,0), 3)


# Display the image with the detections
fig = plt.figure(figsize = (8,8))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('Noisy Image with Face Detections')
ax1.imshow(image_with_detections)
```

```
Number of faces detected: 12
```

```
Out[12]: <matplotlib.image.AxesImage at 0x7fd32e67b780>
```

Step 3: Blur an Image and Perform Edge Detection

Now that we have developed a simple pipeline for detecting faces using OpenCV - let's start playing around with a few fun things we can do with all those detected faces!

Importance of Blur in Edge Detection

Edge detection is a concept that pops up almost everywhere in computer vision applications, as edge-based features (as well as features built on top of edges) are often some of the best features for e.g., object detection and recognition problems.

Edge detection is a dimension reduction technique - by keeping only the edges of an image we get to throw away a lot of non-discriminating information. And typically the most useful kind of edge-detection is one that preserves only the important, global structures (ignoring local structures that aren't very discriminative). So removing local structures / retaining global structures is a crucial pre-processing step to performing edge detection in an image, and blurring can do just that.

Below is an animated gif showing the result of an edge-detected cat taken from Wikipedia (https://en.wikipedia.org/wiki/Gaussian_blur#Common_uses), where the image is gradually blurred more and more prior to edge detection. When the animation begins you can't quite make out what it's a picture of, but as the animation evolves and local structures are removed via blurring the cat becomes visible in the edge-detected image.

Edge detection is a **convolution** performed on the image itself, and you can read about Canny edge detection on this OpenCV documentation page (http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html).

Canny edge detection

In the cell below we load in a test image, then apply *Canny edge detection* on it. The original image is shown on the left panel of the figure, while the edge-detected version of the image is shown on the right. Notice how the result looks very busy - there are too many little details preserved in the image before it is sent to the edge detector. When applied in computer vision applications, edge detection should preserve *global* structure; doing away with local structures that don't help describe what objects are in the image.

In [13]:
```
# Load in the image
image = cv2.imread('images/fawzia.jpg')

# Convert to RGB colorspace
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

# Perform Canny edge detection
edges = cv2.Canny(gray,100,200)

# Dilate the image to amplify edges
edges = cv2.dilate(edges, None)

# Plot the RGB and edge-detected image
fig = plt.figure(figsize = (15,15))
ax1 = fig.add_subplot(121)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('Original Image')
ax1.imshow(image)

ax2 = fig.add_subplot(122)
ax2.set_xticks([])
ax2.set_yticks([])

ax2.set_title('Canny Edges')
ax2.imshow(edges, cmap='gray')
```

Out[13]: <matplotlib.image.AxesImage at 0x7fd32e642710>

Without first blurring the image, and removing small, local structures, a lot of irrelevant edge content gets picked up and amplified by the detector (as shown in the right panel above).

(IMPLEMENTATION) Blur the image *then* perform edge detection

In the next cell, you will repeat this experiment - blurring the image first to remove these local structures, so that only the important boudnary details remain in the edge-detected image.

Blur the image by using OpenCV's `filter2d` functionality - which is discussed in this documentation page (http://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html) - and use an *averaging kernel* of width equal to 4.

In [14]:
```
### TODO: Blur the test imageusing OpenCV's filter2d functionality,
# Use an averaging kernel, and a kernel width equal to 4
# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
blur = cv2.blur(gray,(4,4))

## TODO: Then perform Canny edge detection and display the output

# Perform Canny edge detection
edges = cv2.Canny(blur,100,200)

# Dilate the image to amplify edges
edges = cv2.dilate(edges, None)

# Plot the RGB and edge-detected image
fig = plt.figure(figsize = (15,15))
ax1 = fig.add_subplot(121)
ax1.set_xticks([])
ax1.set_yticks([])

ax1.set_title('Original Image')
ax1.imshow(image)

ax2 = fig.add_subplot(122)
ax2.set_xticks([])
ax2.set_yticks([])

ax2.set_title('Canny Edges')
ax2.imshow(edges, cmap='gray')
```

Out[14]:
```
<matplotlib.image.AxesImage at 0x7fd32e596f28>
```

Step 4: Automatically Hide the Identity of an Individual

If you film something like a documentary or reality TV, you must get permission from every individual shown on film before you can show their face, otherwise you need to blur it out - by blurring the face a lot (so much so that even the global structures are obscured)! This is also true for projects like Google's StreetView maps (https://www.google.com/streetview/) - an enormous collection of mapping images taken from a fleet of Google vehicles. Because it would be impossible for Google to get the permission of every single person accidentally captured in one of these images they blur out everyone's faces, the detected images must automatically blur the identity of detected people. Here's a few examples of folks caught in the camera of a Google street view vehicle.

Read in an image to perform identity detection

Let's try this out for ourselves. Use the face detection pipeline built above and what you know about using the `filter2D` to blur and image, and use these in tandem to hide the identity of the person in the following image - loaded in and printed in the next cell.

In [15]:
```
# Load in the image
image = cv2.imread('images/gus.jpg')

# Convert the image to RGB colorspace
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Display the image
fig = plt.figure(figsize = (6,6))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.set_title('Original Image')
ax1.imshow(image)
```

Out[15]:
```
<matplotlib.image.AxesImage at 0x7fd32e5c0a90>
```

(IMPLEMENTATION) Use blurring to hide the identity of an individual in an image

The idea here is to 1) automatically detect the face in this image, and then 2) blur it out! Make sure to adjust the parameters of the *averaging* blur filter to completely obscure this person's identity.

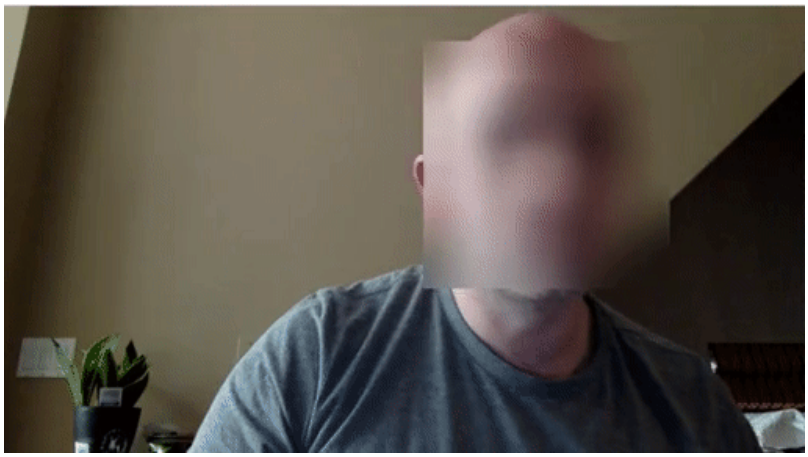
In [16]:
```
## TODO: Implement face detection

## TODO: Blur the bounding box around each detected face using an averaging fi
lter and display the result

def detect_face_and_blur(frame):
    frame_copy=np.copy(frame)
    #originally needed to denoise the frame to find a face well.
    #denoised_frame = cv2.fastNlMeansDenoisingColored(np.copy(frame_copy),Non
e,20,10,7,21)
    gray_frame = cv2.cvtColor(frame_copy, cv2.COLOR_RGB2GRAY)
    #Blurring the gray frame worked substantially faster than denoise
    blurred_gray_frame = cv2.blur(gray_frame,(30,30))
    faces = face_cascade.detectMultiScale(blurred_gray_frame, 1.1, 10)


    # Draw a blur for the faces
    for (x, y, w, h) in faces:
        #cv2.rectangle(frame_copy, (x, y), (x+w, y+h), (0, 255, 0), 2) # to hi
ghlight the face areas for debugging
        frame_copy[y:y+h,x:x+w] = cv2.blur(frame_copy[y:y+h,x:x+w],(60,60)) #m
oar blurry!

    return frame_copy
```

In [17]:
```
image_blurred_face=detect_face_and_blur(image)

# Display the image
fig = plt.figure(figsize = (10,10))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.set_title('Original Image')
ax1.imshow(image_blurred_face)
```

Out[17]: <matplotlib.image.AxesImage at 0x7fd32e571358>

(Optional) Build identity protection into your laptop camera

In this optional task you can add identity protection to your laptop camera, using the previously completed code where you added face detection to your laptop camera - and the task above. You should be able to get reasonable results with little parameter tuning - like the one shown in the gif below.



As with the previous video task, to make this perfect would require significant effort - so don't strive for perfection here, strive for reasonable quality.

The next cell contains code a wrapper function called `laptop_camera_identity_hider` that - when called - will activate your laptop's camera. You need to place the relevant face detection and blurring code developed above in this function in order to blur faces entering your laptop camera's field of view.

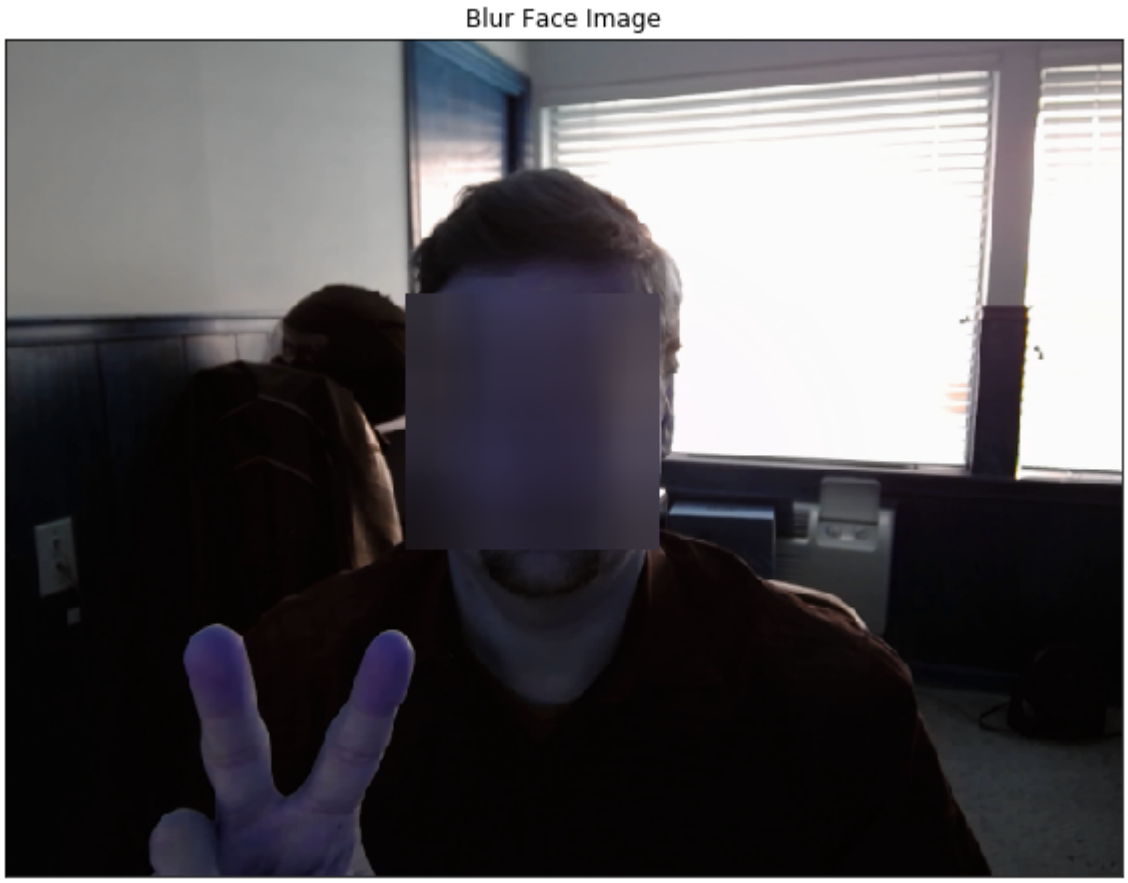
Before adding anything to the function you can call it to get a hang of how it works - a small window will pop up showing you the live feed from your camera, you can press any key to close this window.

Note: Mac users may find that activating this function kills the kernel of their notebook every once in a while. If this happens to you, just restart your notebook's kernel, activate cell(s) containing any crucial import statements, and you'll be good to go!

In [412]:
```
### Insert face detection and blurring code into the wrapper below to create a
n identity protector on your laptop!
import cv2
import time

def laptop_camera_go():
    # Create instance of video capturer
    cv2.namedWindow("face detection activated")
    vc = cv2.VideoCapture(1)

    # Try to get the first frame
    if vc.isOpened():
        rval, frame = vc.read()
    else:
        rval = False

    # Keep video stream open
    while rval:
        # Plot image from camera with detections marked
        sh_image=detect_face_and_blur(frame)
        cv2.imshow("face detection activated", sh_image)

        # Exit functionality - press any key to exit laptop video
        key = cv2.waitKey(20)
        if key > 0: # Exit by pressing any key
            # Destroy windows
            cv2.destroyAllWindows()

            for i in range (1,5):
                cv2.waitKey(1)
            return sh_image

        # Read next frame
        time.sleep(0.05)             # control framerate for computation - def
ault 20 frames per sec
        rval, frame = vc.read()
```

In [413]:
```
# Run laptop identity hider
last_frame=laptop_camera_go()

# Display the image
fig = plt.figure(figsize = (10,10))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.set_title('Blur Face Image')
ax1.imshow( last_frame)
```

Out[413]: <matplotlib.image.AxesImage at 0x205fe8f8208>



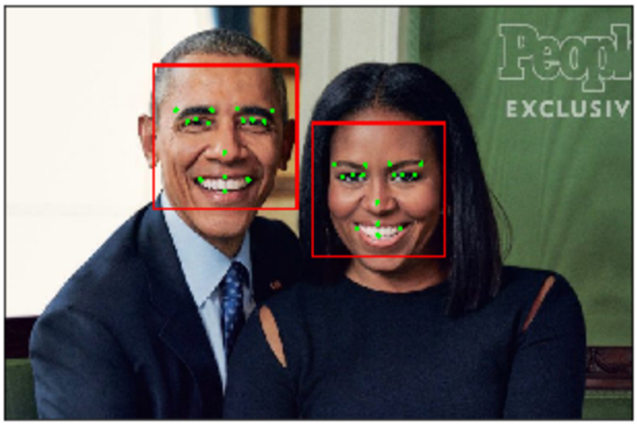
Step 5: Create a CNN to Recognize Facial Keypoints

OpenCV is often used in practice with other machine learning and deep learning libraries to produce interesting results. In this stage of the project you will create your own end-to-end pipeline - employing convolutional networks in keras along with OpenCV - to apply a "selfie" filter to streaming video and images.

You will start by creating and then training a convolutional network that can detect facial keypoints in a small dataset of cropped images of human faces. We then guide you towards OpenCV to expanding your detection algorithm to more general images. What are facial keypoints? Let's take a look at some examples.

Facial keypoints (also called facial landmarks) are the small blue-green dots shown on each of the faces in the image above - there are 15 keypoints marked in each image. They mark important areas of the face - the eyes, corners of the mouth, the nose, etc. Facial keypoints can be used in a variety of machine learning applications from face and emotion recognition to commercial applications like the image filters popularized by Snapchat.

Below we illustrate a filter that, using the results of this section, automatically places sunglasses on people in images (using the facial keypoints to place the glasses correctly on each face). Here, the facial keypoints have been colored lime green for visualization purposes.



Make a facial keypoint detector

But first things first: how can we make a facial keypoint detector? Well, at a high level, notice that facial keypoint detection is a *regression problem*. A single face corresponds to a set of 15 facial keypoints (a set of 15 corresponding (x, y) coordinates, i.e., an output point). Because our input data are images, we can employ a *convolutional neural network* to recognize patterns in our images and learn how to identify these keypoint given sets of labeled data.

In order to train a regressor, we need a training set - a set of facial image / facial keypoint pairs to train on. For this we will be using this dataset from Kaggle (https://www.kaggle.com/c/facial-keypoints-detection/data). We've already downloaded this data and placed it in the `data` directory. Make sure that you have both the *training* and *test* data files. The training dataset contains several thousand 96×96 grayscale images of cropped human faces, along with each face's 15 corresponding facial keypoints (also called landmarks) that have been placed by hand, and recorded in (x, y) coordinates. This wonderful resource also has a substantial testing set, which we will use in tinkering with our convolutional network.

To load in this data, run the Python cell below - notice we will load in both the training and testing sets.

The `load_data` function is in the included `utils.py` file.

In [8]:
```
from utils import *

# Load training set
X_train, y_train = load_data()
print("X_train.shape == {}".format(X_train.shape))
print("y_train.shape == {}; y_train.min == {:.3f}; y_train.max == {:.3f}".form
at(
    y_train.shape, y_train.min(), y_train.max()))

# Load testing set
X_test, _ = load_data(test=True)
print("X_test.shape == {}".format(X_test.shape))
```

```
Using TensorFlow backend.

X_train.shape == (2140, 96, 96, 1)
y_train.shape == (2140, 30); y_train.min == -0.920; y_train.max == 0.996
X_test.shape == (1783, 96, 96, 1)
```

The `load_data` function in `utils.py` originates from this excellent blog post (http://danielnouri.org/notes/2014/12/17/using-convolutional-neural-nets-to-detect-facial-keypoints-tutorial/), which you are *strongly* encouraged to read. Please take the time now to review this function. Note how the output values - that is, the coordinates of each set of facial landmarks - have been normalized to take on values in the range $[-1, 1]$, while the pixel values of each input point (a facial image) have been normalized to the range $[0, 1]$.

Note: the original Kaggle dataset contains some images with several missing keypoints. For simplicity, the `load_data` function removes those images with missing labels from the dataset. As an ***optional*** extension, you are welcome to amend the `load_data` function to include the incomplete data points.

Visualize the Training Data

Execute the code cell below to visualize a subset of the training data.

In [9]:
```
import matplotlib.pyplot as plt
%matplotlib inline

fig = plt.figure(figsize=(20,20))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05
)
for i in range(9):
    ax = fig.add_subplot(3, 3, i + 1, xticks=[], yticks=[])
    plot_data(X_train[i], y_train[i], ax)
```

For each training image, there are two landmarks per eyebrow (**four** total), three per eye (**six** total), **four** for the mouth, and **one** for the tip of the nose.

Review the `plot_data` function in `utils.py` to understand how the 30-dimensional training labels in `y_train` are mapped to facial locations, as this function will prove useful for your pipeline.

(IMPLEMENTATION) Specify the CNN Architecture

In this section, you will specify a neural network for predicting the locations of facial keypoints. Use the code cell below to specify the architecture of your neural network. We have imported some layers that you may find useful for this task, but if you need to use more Keras layers, feel free to import them in the cell.

Your network should accept a 96×96 grayscale image as input, and it should output a vector with 30 entries, corresponding to the predicted (horizontal and vertical) locations of 15 facial keypoints. If you are not sure where to start, you can find some useful starting architectures in this blog (http://danielnouri.org/notes/2014/12/17/using-convolutional-neural-nets-to-detect-facial-keypoints-tutorial/), but you are not permitted to copy any of the architectures that you find online.

In [265]:
```
# Import deep learning resources from Keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout
from keras.layers import Flatten, Dense


## TODO: Specify a CNN architecture
# Your model should accept 96x96 pixel graysale images in
# It should have a fully-connected output layer with 30 values (2 for each fac
ial keypoint)
def base_model(summary=False):
    model = Sequential()
    model.add(Conv2D(filters=32, kernel_size=(3,3), padding='same', activation
='relu',
                              input_shape=(96, 96, 1)))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(filters=64, kernel_size=(3,3), padding='same', activation
='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(filters=128, kernel_size=(2,2), padding='same', activatio
n='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Dropout(0.3))
    model.add(Flatten())
    model.add(Dense(500, activation='relu'))
    model.add(Dropout(0.4))
    model.add(Dense(30))



    # Summarize the model
    if summary==True:
        model.summary()

    return model
base_model(summary=True)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 96, 96, 32)        320
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 48, 48, 64)        18496
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 24, 24, 128)       32896
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
dropout_9 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
flatten_5 (Flatten)          (None, 18432)             0
_________________________________________________________________
dense_9 (Dense)              (None, 500)               9216500
_________________________________________________________________
dropout_10 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_10 (Dense)             (None, 30)                15030
=================================================================
Total params: 9,283,242.0
Trainable params: 9,283,242.0
Non-trainable params: 0.0
_________________________________________________________________
```

Out[265]: <keras.models.Sequential at 0x205bc4204a8>

In [266]:
```
def bigger_base_model(summary=False):
    model = Sequential()
    model.add(Conv2D(filters=32, kernel_size=(5,5), padding='same', activation
='relu',
                              input_shape=(96, 96, 1)))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(filters=64, kernel_size=(3,3), padding='same', activation
='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(filters=128, kernel_size=(2,2), padding='same', activatio
n='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))



    model.add(Dropout(0.3))
    model.add(Flatten())
    model.add(Dense(500, activation='relu'))
    model.add(Dropout(0.4))
    model.add(Dense(30))



    # Summarize the model
    if summary==True:
        model.summary()

    return model
base_model(summary=True)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 96, 96, 32)        320
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 48, 48, 64)        18496
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 24, 24, 128)       32896
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
dropout_11 (Dropout)         (None, 12, 12, 128)       0
_________________________________________________________________
flatten_6 (Flatten)          (None, 18432)             0
_________________________________________________________________
dense_11 (Dense)             (None, 500)               9216500
_________________________________________________________________
dropout_12 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_12 (Dense)             (None, 30)                15030
=================================================================
Total params: 9,283,242.0
Trainable params: 9,283,242.0
Non-trainable params: 0.0
_________________________________________________________________
```

Out[266]: <keras.models.Sequential at 0x205bc4ced68>

In [267]:
```
def dropout_base_model(summary=False):
    model = Sequential()
    model.add(Conv2D(filters=16, kernel_size=(3,3), padding='same', activation
='relu',
                              input_shape=(96, 96, 1)))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.1))

    model.add(Conv2D(filters=32, kernel_size=(2,2), padding='same', activation
='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(filters=64, kernel_size=(2,2), padding='same', activation
='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.3))

    model.add(Flatten())
    model.add(Dense(500, activation='relu'))
    model.add(Dropout(0.4))
    model.add(Dense(30))



    # Summarize the model
    if summary==True:
        model.summary()

    return model
dropout_base_model(summary=True)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_19 (Conv2D)           (None, 96, 96, 16)        160
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 48, 48, 16)        0
_________________________________________________________________
dropout_13 (Dropout)         (None, 48, 48, 16)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 48, 48, 32)        2080
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 24, 24, 32)        0
_________________________________________________________________
dropout_14 (Dropout)         (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 24, 24, 64)        8256
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 12, 12, 64)        0
_________________________________________________________________
dropout_15 (Dropout)         (None, 12, 12, 64)        0
_________________________________________________________________
flatten_7 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_13 (Dense)             (None, 500)               4608500
_________________________________________________________________
dropout_16 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_14 (Dense)             (None, 30)                15030
=================================================================
Total params: 4,634,026.0
Trainable params: 4,634,026.0
Non-trainable params: 0.0
_________________________________________________________________
```

Out[267]: <keras.models.Sequential at 0x205bc672c88>

Step 6: Compile and Train the Model

After specifying your architecture, you'll need to compile and train the model to detect facial keypoints'

(IMPLEMENTATION) Compile and Train the Model

Use the `compile` method (https://keras.io/models/sequential/#sequential-model-methods) to configure the learning process. Experiment with your choice of optimizer (https://keras.io/optimizers/); you may have some ideas about which will work best (`SGD` vs. `RMSprop`, etc), but take the time to empirically verify your theories.

Use the `fit` method (https://keras.io/models/sequential/#sequential-model-methods) to train the model. Break off a validation set by setting `validation_split=0.2`. Save the returned `History` object in the `history` variable.

Your model is required to attain a validation loss (measured as mean squared error) of at least **XYZ**. When you have finished training, save your model (https://keras.io/getting-started/faq/#how-can-i-save-a-keras-model) as an HDF5 file with file path `my_model.h5`.

In [378]:
```
from keras.optimizers import SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Na
dam
from keras.callbacks import ModelCheckpoint

model_path='./model/'
optimizers=['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam'
] #list of optimizers to use

model_names=['base_model', "bigger_base_model",'dropout_base_model']
models=[base_model,bigger_base_model,dropout_base_model]

def run_model(model_func,optimizer):
    model=model_func()
    ## TODO: Compile the model
    model.compile(loss='mean_squared_error', optimizer=optimizer,
                      metrics=['accuracy','mse'])

    # save the best version of the model
    model_weights_filepath=model_path+optimizer+'_model.weights.best.hdf5'
    checkpointer = ModelCheckpoint(filepath=model_weights_filepath, verbose=1,

                                   save_best_only=True)

     ## TODO: Train the model
    hist = model.fit(X_train, y_train, batch_size=32, epochs=50,
              validation_split=0.2, callbacks=[checkpointer],
              verbose=2, shuffle=True)

    # load best weights from training
    model.load_weights(model_weights_filepath)

    ## TODO: Save the model as model.h5
    model.save(model_path+optimizer+'_model.h5')
    return hist,model
```

In [28]:
```
# Train a model and preserve it's history. each model and it's history are sto
red with a key relating to the model and
# optimizer, so that it may be regenerated later
trained_models={}
for i in range(len(models)):
    for opt in optimizers:
        print("\n\n\n  Running model:",model_names[i],"w/opt:",opt)
        trained_models[model_names[i]+'-'+opt]=run_model(models[i],opt)
```

```
  Running model: base_model w/opt: SGD
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.01482, saving model to ./mode
l/SGD_model.weights.best.hdf5
3s - loss: 0.0809 - acc: 0.1577 - mean_squared_error: 0.0809 - val_loss: 0.
0148 - val_acc: 0.6776 - val_mean_squared_error: 0.0148
Epoch 2/50
Epoch 00001: val_loss improved from 0.01482 to 0.01027, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0282 - acc: 0.3306 - mean_squared_error: 0.0282 - val_loss: 0.
0103 - val_acc: 0.6963 - val_mean_squared_error: 0.0103
Epoch 3/50
Epoch 00002: val_loss improved from 0.01027 to 0.00928, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0250 - acc: 0.3680 - mean_squared_error: 0.0250 - val_loss: 0.
0093 - val_acc: 0.6939 - val_mean_squared_error: 0.0093
Epoch 4/50
Epoch 00003: val_loss improved from 0.00928 to 0.00902, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0233 - acc: 0.3879 - mean_squared_error: 0.0233 - val_loss: 0.
0090 - val_acc: 0.6963 - val_mean_squared_error: 0.0090
Epoch 5/50
Epoch 00004: val_loss improved from 0.00902 to 0.00866, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0218 - acc: 0.4182 - mean_squared_error: 0.0218 - val_loss: 0.
0087 - val_acc: 0.6963 - val_mean_squared_error: 0.0087
Epoch 6/50
Epoch 00005: val_loss improved from 0.00866 to 0.00802, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0207 - acc: 0.4036 - mean_squared_error: 0.0207 - val_loss: 0.
0080 - val_acc: 0.6963 - val_mean_squared_error: 0.0080
Epoch 7/50
Epoch 00006: val_loss improved from 0.00802 to 0.00776, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0195 - acc: 0.4206 - mean_squared_error: 0.0195 - val_loss: 0.
0078 - val_acc: 0.6963 - val_mean_squared_error: 0.0078
Epoch 8/50
Epoch 00007: val_loss improved from 0.00776 to 0.00757, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0187 - acc: 0.4416 - mean_squared_error: 0.0187 - val_loss: 0.
0076 - val_acc: 0.6963 - val_mean_squared_error: 0.0076
Epoch 9/50
Epoch 00008: val_loss improved from 0.00757 to 0.00711, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0178 - acc: 0.4550 - mean_squared_error: 0.0178 - val_loss: 0.
0071 - val_acc: 0.6963 - val_mean_squared_error: 0.0071
Epoch 10/50
Epoch 00009: val_loss improved from 0.00711 to 0.00699, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0173 - acc: 0.4644 - mean_squared_error: 0.0173 - val_loss: 0.
0070 - val_acc: 0.6963 - val_mean_squared_error: 0.0070
Epoch 11/50
Epoch 00010: val_loss did not improve
2s - loss: 0.0168 - acc: 0.4428 - mean_squared_error: 0.0168 - val_loss: 0.
```

```
0070 - val_acc: 0.6963 - val_mean_squared_error: 0.0070
Epoch 12/50
Epoch 00011: val_loss improved from 0.00699 to 0.00694, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0162 - acc: 0.4591 - mean_squared_error: 0.0162 - val_loss: 0.
0069 - val_acc: 0.6963 - val_mean_squared_error: 0.0069
Epoch 13/50
Epoch 00012: val_loss improved from 0.00694 to 0.00646, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0157 - acc: 0.4620 - mean_squared_error: 0.0157 - val_loss: 0.
0065 - val_acc: 0.6963 - val_mean_squared_error: 0.0065
Epoch 14/50
Epoch 00013: val_loss did not improve
2s - loss: 0.0156 - acc: 0.4743 - mean_squared_error: 0.0156 - val_loss: 0.
0066 - val_acc: 0.6963 - val_mean_squared_error: 0.0066
Epoch 15/50
Epoch 00014: val_loss improved from 0.00646 to 0.00629, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0149 - acc: 0.4778 - mean_squared_error: 0.0149 - val_loss: 0.
0063 - val_acc: 0.6963 - val_mean_squared_error: 0.0063
Epoch 16/50
Epoch 00015: val_loss improved from 0.00629 to 0.00602, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0146 - acc: 0.4842 - mean_squared_error: 0.0146 - val_loss: 0.
0060 - val_acc: 0.6963 - val_mean_squared_error: 0.0060
Epoch 17/50
Epoch 00016: val_loss improved from 0.00602 to 0.00598, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0142 - acc: 0.4655 - mean_squared_error: 0.0142 - val_loss: 0.
0060 - val_acc: 0.6963 - val_mean_squared_error: 0.0060
Epoch 18/50
Epoch 00017: val_loss did not improve
2s - loss: 0.0139 - acc: 0.4994 - mean_squared_error: 0.0139 - val_loss: 0.
0060 - val_acc: 0.6963 - val_mean_squared_error: 0.0060
Epoch 19/50
Epoch 00018: val_loss improved from 0.00598 to 0.00584, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0138 - acc: 0.5187 - mean_squared_error: 0.0138 - val_loss: 0.
0058 - val_acc: 0.6963 - val_mean_squared_error: 0.0058
Epoch 20/50
Epoch 00019: val_loss improved from 0.00584 to 0.00569, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0135 - acc: 0.5012 - mean_squared_error: 0.0135 - val_loss: 0.
0057 - val_acc: 0.6963 - val_mean_squared_error: 0.0057
Epoch 21/50
Epoch 00020: val_loss did not improve
2s - loss: 0.0132 - acc: 0.5117 - mean_squared_error: 0.0132 - val_loss: 0.
0059 - val_acc: 0.6963 - val_mean_squared_error: 0.0059
Epoch 22/50
Epoch 00021: val_loss improved from 0.00569 to 0.00544, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0131 - acc: 0.5029 - mean_squared_error: 0.0131 - val_loss: 0.
0054 - val_acc: 0.6963 - val_mean_squared_error: 0.0054
Epoch 23/50
Epoch 00022: val_loss did not improve
2s - loss: 0.0127 - acc: 0.4907 - mean_squared_error: 0.0127 - val_loss: 0.
0055 - val_acc: 0.6963 - val_mean_squared_error: 0.0055
```

```
Epoch 24/50
Epoch 00023: val_loss did not improve
2s - loss: 0.0124 - acc: 0.5374 - mean_squared_error: 0.0124 - val_loss: 0.
0055 - val_acc: 0.6963 - val_mean_squared_error: 0.0055
Epoch 25/50
Epoch 00024: val_loss improved from 0.00544 to 0.00542, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0124 - acc: 0.5134 - mean_squared_error: 0.0124 - val_loss: 0.
0054 - val_acc: 0.6963 - val_mean_squared_error: 0.0054
Epoch 26/50
Epoch 00025: val_loss improved from 0.00542 to 0.00530, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0122 - acc: 0.5076 - mean_squared_error: 0.0122 - val_loss: 0.
0053 - val_acc: 0.6963 - val_mean_squared_error: 0.0053
Epoch 27/50
Epoch 00026: val_loss improved from 0.00530 to 0.00518, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0120 - acc: 0.5134 - mean_squared_error: 0.0120 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 28/50
Epoch 00027: val_loss did not improve
2s - loss: 0.0119 - acc: 0.5181 - mean_squared_error: 0.0119 - val_loss: 0.
0053 - val_acc: 0.6963 - val_mean_squared_error: 0.0053
Epoch 29/50
Epoch 00028: val_loss improved from 0.00518 to 0.00517, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0119 - acc: 0.5082 - mean_squared_error: 0.0119 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 30/50
Epoch 00029: val_loss did not improve
2s - loss: 0.0117 - acc: 0.5146 - mean_squared_error: 0.0117 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0115 - acc: 0.5152 - mean_squared_error: 0.0115 - val_loss: 0.
0053 - val_acc: 0.6963 - val_mean_squared_error: 0.0053
Epoch 32/50
Epoch 00031: val_loss improved from 0.00517 to 0.00504, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0115 - acc: 0.5473 - mean_squared_error: 0.0115 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 33/50
Epoch 00032: val_loss did not improve
2s - loss: 0.0112 - acc: 0.5164 - mean_squared_error: 0.0112 - val_loss: 0.
0051 - val_acc: 0.6963 - val_mean_squared_error: 0.0051
Epoch 34/50
Epoch 00033: val_loss improved from 0.00504 to 0.00499, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0111 - acc: 0.5456 - mean_squared_error: 0.0111 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0111 - acc: 0.5082 - mean_squared_error: 0.0111 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0109 - acc: 0.5146 - mean_squared_error: 0.0109 - val_loss: 0.
```

```
0051 - val_acc: 0.6963 - val_mean_squared_error: 0.0051
Epoch 37/50
Epoch 00036: val_loss improved from 0.00499 to 0.00491, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0108 - acc: 0.5315 - mean_squared_error: 0.0108 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 38/50
Epoch 00037: val_loss improved from 0.00491 to 0.00489, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0108 - acc: 0.5345 - mean_squared_error: 0.0108 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 39/50
Epoch 00038: val_loss did not improve
2s - loss: 0.0105 - acc: 0.5386 - mean_squared_error: 0.0105 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 40/50
Epoch 00039: val_loss did not improve
2s - loss: 0.0106 - acc: 0.5193 - mean_squared_error: 0.0106 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 41/50
Epoch 00040: val_loss improved from 0.00489 to 0.00479, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0106 - acc: 0.5561 - mean_squared_error: 0.0106 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 42/50
Epoch 00041: val_loss improved from 0.00479 to 0.00475, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0103 - acc: 0.5543 - mean_squared_error: 0.0103 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 43/50
Epoch 00042: val_loss did not improve
2s - loss: 0.0104 - acc: 0.5549 - mean_squared_error: 0.0104 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0103 - acc: 0.5444 - mean_squared_error: 0.0103 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 45/50
Epoch 00044: val_loss did not improve
2s - loss: 0.0101 - acc: 0.5409 - mean_squared_error: 0.0101 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 46/50
Epoch 00045: val_loss did not improve
2s - loss: 0.0101 - acc: 0.5444 - mean_squared_error: 0.0101 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0101 - acc: 0.5461 - mean_squared_error: 0.0101 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 48/50
Epoch 00047: val_loss did not improve
2s - loss: 0.0101 - acc: 0.5485 - mean_squared_error: 0.0101 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0100 - acc: 0.5502 - mean_squared_error: 0.0100 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
```

```
Epoch 50/50
Epoch 00049: val_loss did not improve
2s - loss: 0.0098 - acc: 0.5660 - mean_squared_error: 0.0098 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049



  Running model: base_model w/opt: RMSprop
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.00879, saving model to ./mode
l/RMSprop_model.weights.best.hdf5
3s - loss: 0.2576 - acc: 0.3697 - mean_squared_error: 0.2576 - val_loss: 0.
0088 - val_acc: 0.5818 - val_mean_squared_error: 0.0088
Epoch 2/50
Epoch 00001: val_loss improved from 0.00879 to 0.00551, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0242 - acc: 0.4655 - mean_squared_error: 0.0242 - val_loss: 0.
0055 - val_acc: 0.6963 - val_mean_squared_error: 0.0055
Epoch 3/50
Epoch 00002: val_loss did not improve
2s - loss: 0.0127 - acc: 0.5245 - mean_squared_error: 0.0127 - val_loss: 0.
0067 - val_acc: 0.6963 - val_mean_squared_error: 0.0067
Epoch 4/50
Epoch 00003: val_loss did not improve
2s - loss: 0.0092 - acc: 0.6092 - mean_squared_error: 0.0092 - val_loss: 0.
0074 - val_acc: 0.6963 - val_mean_squared_error: 0.0074
Epoch 5/50
Epoch 00004: val_loss improved from 0.00551 to 0.00418, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0078 - acc: 0.6244 - mean_squared_error: 0.0078 - val_loss: 0.
0042 - val_acc: 0.6963 - val_mean_squared_error: 0.0042
Epoch 6/50
Epoch 00005: val_loss did not improve
2s - loss: 0.0065 - acc: 0.6636 - mean_squared_error: 0.0065 - val_loss: 0.
0064 - val_acc: 0.6916 - val_mean_squared_error: 0.0064
Epoch 7/50
Epoch 00006: val_loss did not improve
2s - loss: 0.0060 - acc: 0.6711 - mean_squared_error: 0.0060 - val_loss: 0.
0055 - val_acc: 0.6916 - val_mean_squared_error: 0.0055
Epoch 8/50
Epoch 00007: val_loss improved from 0.00418 to 0.00348, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0051 - acc: 0.6869 - mean_squared_error: 0.0051 - val_loss: 0.
0035 - val_acc: 0.6939 - val_mean_squared_error: 0.0035
Epoch 9/50
Epoch 00008: val_loss improved from 0.00348 to 0.00262, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0048 - acc: 0.6875 - mean_squared_error: 0.0048 - val_loss: 0.
0026 - val_acc: 0.7126 - val_mean_squared_error: 0.0026
Epoch 10/50
Epoch 00009: val_loss did not improve
2s - loss: 0.0045 - acc: 0.6963 - mean_squared_error: 0.0045 - val_loss: 0.
0063 - val_acc: 0.7220 - val_mean_squared_error: 0.0063
Epoch 11/50
Epoch 00010: val_loss did not improve
2s - loss: 0.0039 - acc: 0.6974 - mean_squared_error: 0.0039 - val_loss: 0.
```

```
0030 - val_acc: 0.7173 - val_mean_squared_error: 0.0030
Epoch 12/50
Epoch 00011: val_loss improved from 0.00262 to 0.00202, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0037 - acc: 0.7114 - mean_squared_error: 0.0037 - val_loss: 0.
0020 - val_acc: 0.7196 - val_mean_squared_error: 0.0020
Epoch 13/50
Epoch 00012: val_loss did not improve
2s - loss: 0.0034 - acc: 0.7225 - mean_squared_error: 0.0034 - val_loss: 0.
0021 - val_acc: 0.7196 - val_mean_squared_error: 0.0021
Epoch 14/50
Epoch 00013: val_loss improved from 0.00202 to 0.00188, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0032 - acc: 0.7202 - mean_squared_error: 0.0032 - val_loss: 0.
0019 - val_acc: 0.7360 - val_mean_squared_error: 0.0019
Epoch 15/50
Epoch 00014: val_loss did not improve
2s - loss: 0.0029 - acc: 0.7266 - mean_squared_error: 0.0029 - val_loss: 0.
0022 - val_acc: 0.7103 - val_mean_squared_error: 0.0022
Epoch 16/50
Epoch 00015: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7336 - mean_squared_error: 0.0027 - val_loss: 0.
0025 - val_acc: 0.7407 - val_mean_squared_error: 0.0025
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7284 - mean_squared_error: 0.0027 - val_loss: 0.
0022 - val_acc: 0.7430 - val_mean_squared_error: 0.0022
Epoch 18/50
Epoch 00017: val_loss did not improve
2s - loss: 0.0023 - acc: 0.7237 - mean_squared_error: 0.0023 - val_loss: 0.
0024 - val_acc: 0.7313 - val_mean_squared_error: 0.0024
Epoch 19/50
Epoch 00018: val_loss improved from 0.00188 to 0.00152, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0024 - acc: 0.7447 - mean_squared_error: 0.0024 - val_loss: 0.
0015 - val_acc: 0.7383 - val_mean_squared_error: 0.0015
Epoch 20/50
Epoch 00019: val_loss did not improve
2s - loss: 0.0021 - acc: 0.7523 - mean_squared_error: 0.0021 - val_loss: 0.
0017 - val_acc: 0.7407 - val_mean_squared_error: 0.0017
Epoch 21/50
Epoch 00020: val_loss improved from 0.00152 to 0.00151, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0021 - acc: 0.7494 - mean_squared_error: 0.0021 - val_loss: 0.
0015 - val_acc: 0.7523 - val_mean_squared_error: 0.0015
Epoch 22/50
Epoch 00021: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7494 - mean_squared_error: 0.0019 - val_loss: 0.
0016 - val_acc: 0.7617 - val_mean_squared_error: 0.0016
Epoch 23/50
Epoch 00022: val_loss improved from 0.00151 to 0.00139, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0019 - acc: 0.7687 - mean_squared_error: 0.0019 - val_loss: 0.
0014 - val_acc: 0.7640 - val_mean_squared_error: 0.0014
Epoch 24/50
Epoch 00023: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7605 - mean_squared_error: 0.0018 - val_loss: 0.
```

```
0015 - val_acc: 0.7593 - val_mean_squared_error: 0.0015
Epoch 25/50
Epoch 00024: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7646 - mean_squared_error: 0.0017 - val_loss: 0.
0019 - val_acc: 0.7290 - val_mean_squared_error: 0.0019
Epoch 26/50
Epoch 00025: val_loss improved from 0.00139 to 0.00134, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0017 - acc: 0.7798 - mean_squared_error: 0.0017 - val_loss: 0.
0013 - val_acc: 0.7523 - val_mean_squared_error: 0.0013
Epoch 27/50
Epoch 00026: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7827 - mean_squared_error: 0.0017 - val_loss: 0.
0016 - val_acc: 0.7780 - val_mean_squared_error: 0.0016
Epoch 28/50
Epoch 00027: val_loss improved from 0.00134 to 0.00129, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0016 - acc: 0.7646 - mean_squared_error: 0.0016 - val_loss: 0.
0013 - val_acc: 0.7570 - val_mean_squared_error: 0.0013
Epoch 29/50
Epoch 00028: val_loss improved from 0.00129 to 0.00125, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0015 - acc: 0.7763 - mean_squared_error: 0.0015 - val_loss: 0.
0012 - val_acc: 0.7664 - val_mean_squared_error: 0.0012
Epoch 30/50
Epoch 00029: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7926 - mean_squared_error: 0.0015 - val_loss: 0.
0013 - val_acc: 0.7687 - val_mean_squared_error: 0.0013
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7862 - mean_squared_error: 0.0015 - val_loss: 0.
0013 - val_acc: 0.7710 - val_mean_squared_error: 0.0013
Epoch 32/50
Epoch 00031: val_loss improved from 0.00125 to 0.00115, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0014 - acc: 0.7915 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7734 - val_mean_squared_error: 0.0012
Epoch 33/50
Epoch 00032: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7880 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7523 - val_mean_squared_error: 0.0012
Epoch 34/50
Epoch 00033: val_loss did not improve
2s - loss: 0.0014 - acc: 0.8061 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7921 - val_mean_squared_error: 0.0012
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7967 - mean_squared_error: 0.0013 - val_loss: 0.
0014 - val_acc: 0.7664 - val_mean_squared_error: 0.0014
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7862 - mean_squared_error: 0.0013 - val_loss: 0.
0014 - val_acc: 0.7547 - val_mean_squared_error: 0.0014
Epoch 37/50
Epoch 00036: val_loss did not improve
2s - loss: 0.0013 - acc: 0.8143 - mean_squared_error: 0.0013 - val_loss: 0.
0012 - val_acc: 0.7850 - val_mean_squared_error: 0.0012
```

```
Epoch 38/50
Epoch 00037: val_loss did not improve
2s - loss: 0.0012 - acc: 0.8154 - mean_squared_error: 0.0012 - val_loss: 0.
0012 - val_acc: 0.7874 - val_mean_squared_error: 0.0012
Epoch 39/50
Epoch 00038: val_loss did not improve
2s - loss: 0.0012 - acc: 0.8107 - mean_squared_error: 0.0012 - val_loss: 0.
0013 - val_acc: 0.7757 - val_mean_squared_error: 0.0013
Epoch 40/50
Epoch 00039: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8090 - mean_squared_error: 0.0011 - val_loss: 0.
0013 - val_acc: 0.7710 - val_mean_squared_error: 0.0013
Epoch 41/50
Epoch 00040: val_loss did not improve
2s - loss: 0.0011 - acc: 0.7950 - mean_squared_error: 0.0011 - val_loss: 0.
0013 - val_acc: 0.8107 - val_mean_squared_error: 0.0013
Epoch 42/50
Epoch 00041: val_loss improved from 0.00115 to 0.00110, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0011 - acc: 0.8248 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.8084 - val_mean_squared_error: 0.0011
Epoch 43/50
Epoch 00042: val_loss improved from 0.00110 to 0.00108, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0011 - acc: 0.8201 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.8131 - val_mean_squared_error: 0.0011
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8096 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.8037 - val_mean_squared_error: 0.0011
Epoch 45/50
Epoch 00044: val_loss improved from 0.00108 to 0.00106, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0010 - acc: 0.8213 - mean_squared_error: 0.0010 - val_loss: 0.
0011 - val_acc: 0.7944 - val_mean_squared_error: 0.0011
Epoch 46/50
Epoch 00045: val_loss did not improve
2s - loss: 9.9709e-04 - acc: 0.8166 - mean_squared_error: 9.9709e-04 - val_
loss: 0.0015 - val_acc: 0.8224 - val_mean_squared_error: 0.0015
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0010 - acc: 0.8148 - mean_squared_error: 0.0010 - val_loss: 0.
0011 - val_acc: 0.7944 - val_mean_squared_error: 0.0011
Epoch 48/50
Epoch 00047: val_loss did not improve
2s - loss: 9.9569e-04 - acc: 0.8154 - mean_squared_error: 9.9569e-04 - val_
loss: 0.0011 - val_acc: 0.8084 - val_mean_squared_error: 0.0011
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 9.6123e-04 - acc: 0.8084 - mean_squared_error: 9.6123e-04 - val_
loss: 0.0011 - val_acc: 0.8037 - val_mean_squared_error: 0.0011
Epoch 50/50
Epoch 00049: val_loss did not improve
2s - loss: 9.3249e-04 - acc: 0.8131 - mean_squared_error: 9.3249e-04 - val_
loss: 0.0012 - val_acc: 0.7640 - val_mean_squared_error: 0.0012
```

```
  Running model: base_model w/opt: Adagrad
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.00827, saving model to ./mode
l/Adagrad_model.weights.best.hdf5
3s - loss: 65.7618 - acc: 0.3873 - mean_squared_error: 65.7618 - val_loss:
0.0083 - val_acc: 0.6963 - val_mean_squared_error: 0.0083
Epoch 2/50
Epoch 00001: val_loss did not improve
2s - loss: 0.0205 - acc: 0.4375 - mean_squared_error: 0.0205 - val_loss: 0.
0120 - val_acc: 0.5864 - val_mean_squared_error: 0.0120
Epoch 3/50
Epoch 00002: val_loss improved from 0.00827 to 0.00454, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0182 - acc: 0.4749 - mean_squared_error: 0.0182 - val_loss: 0.
0045 - val_acc: 0.6963 - val_mean_squared_error: 0.0045
Epoch 4/50
Epoch 00003: val_loss did not improve
2s - loss: 0.0178 - acc: 0.4352 - mean_squared_error: 0.0178 - val_loss: 0.
0083 - val_acc: 0.6916 - val_mean_squared_error: 0.0083
Epoch 5/50
Epoch 00004: val_loss improved from 0.00454 to 0.00431, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0170 - acc: 0.4597 - mean_squared_error: 0.0170 - val_loss: 0.
0043 - val_acc: 0.6939 - val_mean_squared_error: 0.0043
Epoch 6/50
Epoch 00005: val_loss did not improve
2s - loss: 0.0170 - acc: 0.4574 - mean_squared_error: 0.0170 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 7/50
Epoch 00006: val_loss did not improve
2s - loss: 0.0168 - acc: 0.4796 - mean_squared_error: 0.0168 - val_loss: 0.
0045 - val_acc: 0.6963 - val_mean_squared_error: 0.0045
Epoch 8/50
Epoch 00007: val_loss did not improve
2s - loss: 0.0163 - acc: 0.4766 - mean_squared_error: 0.0163 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 9/50
Epoch 00008: val_loss improved from 0.00431 to 0.00368, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0160 - acc: 0.4556 - mean_squared_error: 0.0160 - val_loss: 0.
0037 - val_acc: 0.7103 - val_mean_squared_error: 0.0037
Epoch 10/50
Epoch 00009: val_loss did not improve
2s - loss: 0.0157 - acc: 0.4568 - mean_squared_error: 0.0157 - val_loss: 0.
0042 - val_acc: 0.6939 - val_mean_squared_error: 0.0042
Epoch 11/50
Epoch 00010: val_loss improved from 0.00368 to 0.00347, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0154 - acc: 0.4755 - mean_squared_error: 0.0154 - val_loss: 0.
0035 - val_acc: 0.6939 - val_mean_squared_error: 0.0035
Epoch 12/50
Epoch 00011: val_loss did not improve
2s - loss: 0.0150 - acc: 0.4918 - mean_squared_error: 0.0150 - val_loss: 0.
0053 - val_acc: 0.7266 - val_mean_squared_error: 0.0053
Epoch 13/50
```

```
Epoch 00012: val_loss did not improve
2s - loss: 0.0151 - acc: 0.4597 - mean_squared_error: 0.0151 - val_loss: 0.
0035 - val_acc: 0.6963 - val_mean_squared_error: 0.0035
Epoch 14/50
Epoch 00013: val_loss improved from 0.00347 to 0.00290, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0146 - acc: 0.4965 - mean_squared_error: 0.0146 - val_loss: 0.
0029 - val_acc: 0.7009 - val_mean_squared_error: 0.0029
Epoch 15/50
Epoch 00014: val_loss did not improve
2s - loss: 0.0146 - acc: 0.4766 - mean_squared_error: 0.0146 - val_loss: 0.
0040 - val_acc: 0.6986 - val_mean_squared_error: 0.0040
Epoch 16/50
Epoch 00015: val_loss did not improve
2s - loss: 0.0139 - acc: 0.4825 - mean_squared_error: 0.0139 - val_loss: 0.
0046 - val_acc: 0.7009 - val_mean_squared_error: 0.0046
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0139 - acc: 0.4807 - mean_squared_error: 0.0139 - val_loss: 0.
0030 - val_acc: 0.6939 - val_mean_squared_error: 0.0030
Epoch 18/50
Epoch 00017: val_loss did not improve
2s - loss: 0.0136 - acc: 0.4737 - mean_squared_error: 0.0136 - val_loss: 0.
0047 - val_acc: 0.7523 - val_mean_squared_error: 0.0047
Epoch 19/50
Epoch 00018: val_loss did not improve
2s - loss: 0.0137 - acc: 0.4813 - mean_squared_error: 0.0137 - val_loss: 0.
0039 - val_acc: 0.7056 - val_mean_squared_error: 0.0039
Epoch 20/50
Epoch 00019: val_loss did not improve
2s - loss: 0.0134 - acc: 0.4877 - mean_squared_error: 0.0134 - val_loss: 0.
0034 - val_acc: 0.7033 - val_mean_squared_error: 0.0034
Epoch 21/50
Epoch 00020: val_loss improved from 0.00290 to 0.00273, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0133 - acc: 0.4924 - mean_squared_error: 0.0133 - val_loss: 0.
0027 - val_acc: 0.7103 - val_mean_squared_error: 0.0027
Epoch 22/50
Epoch 00021: val_loss did not improve
2s - loss: 0.0130 - acc: 0.4907 - mean_squared_error: 0.0130 - val_loss: 0.
0041 - val_acc: 0.7150 - val_mean_squared_error: 0.0041
Epoch 23/50
Epoch 00022: val_loss improved from 0.00273 to 0.00262, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0131 - acc: 0.4994 - mean_squared_error: 0.0131 - val_loss: 0.
0026 - val_acc: 0.7477 - val_mean_squared_error: 0.0026
Epoch 24/50
Epoch 00023: val_loss improved from 0.00262 to 0.00240, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0128 - acc: 0.4895 - mean_squared_error: 0.0128 - val_loss: 0.
0024 - val_acc: 0.7453 - val_mean_squared_error: 0.0024
Epoch 25/50
Epoch 00024: val_loss did not improve
2s - loss: 0.0127 - acc: 0.5023 - mean_squared_error: 0.0127 - val_loss: 0.
0033 - val_acc: 0.7313 - val_mean_squared_error: 0.0033
Epoch 26/50
Epoch 00025: val_loss improved from 0.00240 to 0.00223, saving model to ./m
```

```
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0124 - acc: 0.5053 - mean_squared_error: 0.0124 - val_loss: 0.
0022 - val_acc: 0.7103 - val_mean_squared_error: 0.0022
Epoch 27/50
Epoch 00026: val_loss did not improve
2s - loss: 0.0121 - acc: 0.4912 - mean_squared_error: 0.0121 - val_loss: 0.
0047 - val_acc: 0.7173 - val_mean_squared_error: 0.0047
Epoch 28/50
Epoch 00027: val_loss did not improve
2s - loss: 0.0123 - acc: 0.4994 - mean_squared_error: 0.0123 - val_loss: 0.
0032 - val_acc: 0.7266 - val_mean_squared_error: 0.0032
Epoch 29/50
Epoch 00028: val_loss did not improve
2s - loss: 0.0119 - acc: 0.5169 - mean_squared_error: 0.0119 - val_loss: 0.
0027 - val_acc: 0.7173 - val_mean_squared_error: 0.0027
Epoch 30/50
Epoch 00029: val_loss did not improve
2s - loss: 0.0118 - acc: 0.5035 - mean_squared_error: 0.0118 - val_loss: 0.
0030 - val_acc: 0.7079 - val_mean_squared_error: 0.0030
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0118 - acc: 0.4942 - mean_squared_error: 0.0118 - val_loss: 0.
0031 - val_acc: 0.7196 - val_mean_squared_error: 0.0031
Epoch 32/50
Epoch 00031: val_loss did not improve
2s - loss: 0.0118 - acc: 0.4988 - mean_squared_error: 0.0118 - val_loss: 0.
0041 - val_acc: 0.7196 - val_mean_squared_error: 0.0041
Epoch 33/50
Epoch 00032: val_loss did not improve
2s - loss: 0.0117 - acc: 0.5134 - mean_squared_error: 0.0117 - val_loss: 0.
0024 - val_acc: 0.7150 - val_mean_squared_error: 0.0024
Epoch 34/50
Epoch 00033: val_loss did not improve
2s - loss: 0.0115 - acc: 0.5053 - mean_squared_error: 0.0115 - val_loss: 0.
0030 - val_acc: 0.7196 - val_mean_squared_error: 0.0030
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0113 - acc: 0.5012 - mean_squared_error: 0.0113 - val_loss: 0.
0028 - val_acc: 0.7453 - val_mean_squared_error: 0.0028
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0113 - acc: 0.5199 - mean_squared_error: 0.0113 - val_loss: 0.
0034 - val_acc: 0.7150 - val_mean_squared_error: 0.0034
Epoch 37/50
Epoch 00036: val_loss did not improve
2s - loss: 0.0113 - acc: 0.5280 - mean_squared_error: 0.0113 - val_loss: 0.
0028 - val_acc: 0.7290 - val_mean_squared_error: 0.0028
Epoch 38/50
Epoch 00037: val_loss improved from 0.00223 to 0.00208, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0115 - acc: 0.5088 - mean_squared_error: 0.0115 - val_loss: 0.
0021 - val_acc: 0.7360 - val_mean_squared_error: 0.0021
Epoch 39/50
Epoch 00038: val_loss did not improve
2s - loss: 0.0110 - acc: 0.5199 - mean_squared_error: 0.0110 - val_loss: 0.
0034 - val_acc: 0.7523 - val_mean_squared_error: 0.0034
Epoch 40/50
```

```
Epoch 00039: val_loss did not improve
2s - loss: 0.0112 - acc: 0.5239 - mean_squared_error: 0.0112 - val_loss: 0.
0025 - val_acc: 0.7079 - val_mean_squared_error: 0.0025
Epoch 41/50
Epoch 00040: val_loss improved from 0.00208 to 0.00191, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0109 - acc: 0.5327 - mean_squared_error: 0.0109 - val_loss: 0.
0019 - val_acc: 0.7336 - val_mean_squared_error: 0.0019
Epoch 42/50
Epoch 00041: val_loss did not improve
2s - loss: 0.0109 - acc: 0.5292 - mean_squared_error: 0.0109 - val_loss: 0.
0022 - val_acc: 0.7500 - val_mean_squared_error: 0.0022
Epoch 43/50
Epoch 00042: val_loss did not improve
2s - loss: 0.0104 - acc: 0.5280 - mean_squared_error: 0.0104 - val_loss: 0.
0025 - val_acc: 0.7453 - val_mean_squared_error: 0.0025
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0106 - acc: 0.5590 - mean_squared_error: 0.0106 - val_loss: 0.
0039 - val_acc: 0.7523 - val_mean_squared_error: 0.0039
Epoch 45/50
Epoch 00044: val_loss did not improve
2s - loss: 0.0104 - acc: 0.5082 - mean_squared_error: 0.0104 - val_loss: 0.
0028 - val_acc: 0.7243 - val_mean_squared_error: 0.0028
Epoch 46/50
Epoch 00045: val_loss did not improve
2s - loss: 0.0105 - acc: 0.5380 - mean_squared_error: 0.0105 - val_loss: 0.
0025 - val_acc: 0.7383 - val_mean_squared_error: 0.0025
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0106 - acc: 0.5345 - mean_squared_error: 0.0106 - val_loss: 0.
0022 - val_acc: 0.7290 - val_mean_squared_error: 0.0022
Epoch 48/50
Epoch 00047: val_loss did not improve
2s - loss: 0.0104 - acc: 0.5602 - mean_squared_error: 0.0104 - val_loss: 0.
0024 - val_acc: 0.7266 - val_mean_squared_error: 0.0024
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0104 - acc: 0.5333 - mean_squared_error: 0.0104 - val_loss: 0.
0023 - val_acc: 0.7360 - val_mean_squared_error: 0.0023
Epoch 50/50
Epoch 00049: val_loss did not improve
2s - loss: 0.0100 - acc: 0.5532 - mean_squared_error: 0.0100 - val_loss: 0.
0019 - val_acc: 0.7500 - val_mean_squared_error: 0.0019



  Running model: base_model w/opt: Adadelta
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.00756, saving model to ./mode
l/Adadelta_model.weights.best.hdf5
4s - loss: 0.0286 - acc: 0.4106 - mean_squared_error: 0.0286 - val_loss: 0.
0076 - val_acc: 0.6963 - val_mean_squared_error: 0.0076
Epoch 2/50
Epoch 00001: val_loss improved from 0.00756 to 0.00507, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
```

```
3s - loss: 0.0145 - acc: 0.4988 - mean_squared_error: 0.0145 - val_loss: 0.
0051 - val_acc: 0.6963 - val_mean_squared_error: 0.0051
Epoch 3/50
Epoch 00002: val_loss improved from 0.00507 to 0.00503, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0112 - acc: 0.5315 - mean_squared_error: 0.0112 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 4/50
Epoch 00003: val_loss improved from 0.00503 to 0.00432, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0092 - acc: 0.5514 - mean_squared_error: 0.0092 - val_loss: 0.
0043 - val_acc: 0.6963 - val_mean_squared_error: 0.0043
Epoch 5/50
Epoch 00004: val_loss improved from 0.00432 to 0.00427, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0084 - acc: 0.5882 - mean_squared_error: 0.0084 - val_loss: 0.
0043 - val_acc: 0.6963 - val_mean_squared_error: 0.0043
Epoch 6/50
Epoch 00005: val_loss improved from 0.00427 to 0.00420, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0076 - acc: 0.6104 - mean_squared_error: 0.0076 - val_loss: 0.
0042 - val_acc: 0.6963 - val_mean_squared_error: 0.0042
Epoch 7/50
Epoch 00006: val_loss improved from 0.00420 to 0.00410, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0070 - acc: 0.6215 - mean_squared_error: 0.0070 - val_loss: 0.
0041 - val_acc: 0.6963 - val_mean_squared_error: 0.0041
Epoch 8/50
Epoch 00007: val_loss improved from 0.00410 to 0.00408, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0068 - acc: 0.6279 - mean_squared_error: 0.0068 - val_loss: 0.
0041 - val_acc: 0.6963 - val_mean_squared_error: 0.0041
Epoch 9/50
Epoch 00008: val_loss did not improve
2s - loss: 0.0064 - acc: 0.6221 - mean_squared_error: 0.0064 - val_loss: 0.
0041 - val_acc: 0.6963 - val_mean_squared_error: 0.0041
Epoch 10/50
Epoch 00009: val_loss did not improve
2s - loss: 0.0061 - acc: 0.6565 - mean_squared_error: 0.0061 - val_loss: 0.
0042 - val_acc: 0.6963 - val_mean_squared_error: 0.0042
Epoch 11/50
Epoch 00010: val_loss improved from 0.00408 to 0.00375, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0059 - acc: 0.6659 - mean_squared_error: 0.0059 - val_loss: 0.
0037 - val_acc: 0.6963 - val_mean_squared_error: 0.0037
Epoch 12/50
Epoch 00011: val_loss improved from 0.00375 to 0.00351, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0055 - acc: 0.6682 - mean_squared_error: 0.0055 - val_loss: 0.
0035 - val_acc: 0.6963 - val_mean_squared_error: 0.0035
Epoch 13/50
Epoch 00012: val_loss did not improve
2s - loss: 0.0055 - acc: 0.6793 - mean_squared_error: 0.0055 - val_loss: 0.
0045 - val_acc: 0.6963 - val_mean_squared_error: 0.0045
Epoch 14/50
Epoch 00013: val_loss improved from 0.00351 to 0.00332, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
```

```
3s - loss: 0.0053 - acc: 0.6916 - mean_squared_error: 0.0053 - val_loss: 0.
0033 - val_acc: 0.6963 - val_mean_squared_error: 0.0033
Epoch 15/50
Epoch 00014: val_loss improved from 0.00332 to 0.00327, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0049 - acc: 0.6776 - mean_squared_error: 0.0049 - val_loss: 0.
0033 - val_acc: 0.6963 - val_mean_squared_error: 0.0033
Epoch 16/50
Epoch 00015: val_loss improved from 0.00327 to 0.00313, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0049 - acc: 0.6811 - mean_squared_error: 0.0049 - val_loss: 0.
0031 - val_acc: 0.7009 - val_mean_squared_error: 0.0031
Epoch 17/50
Epoch 00016: val_loss improved from 0.00313 to 0.00307, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0048 - acc: 0.6811 - mean_squared_error: 0.0048 - val_loss: 0.
0031 - val_acc: 0.6963 - val_mean_squared_error: 0.0031
Epoch 18/50
Epoch 00017: val_loss improved from 0.00307 to 0.00302, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0046 - acc: 0.6922 - mean_squared_error: 0.0046 - val_loss: 0.
0030 - val_acc: 0.6963 - val_mean_squared_error: 0.0030
Epoch 19/50
Epoch 00018: val_loss did not improve
2s - loss: 0.0044 - acc: 0.6735 - mean_squared_error: 0.0044 - val_loss: 0.
0030 - val_acc: 0.6986 - val_mean_squared_error: 0.0030
Epoch 20/50
Epoch 00019: val_loss did not improve
2s - loss: 0.0044 - acc: 0.6974 - mean_squared_error: 0.0044 - val_loss: 0.
0034 - val_acc: 0.7056 - val_mean_squared_error: 0.0034
Epoch 21/50
Epoch 00020: val_loss improved from 0.00302 to 0.00290, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0042 - acc: 0.6887 - mean_squared_error: 0.0042 - val_loss: 0.
0029 - val_acc: 0.6963 - val_mean_squared_error: 0.0029
Epoch 22/50
Epoch 00021: val_loss improved from 0.00290 to 0.00286, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0040 - acc: 0.6875 - mean_squared_error: 0.0040 - val_loss: 0.
0029 - val_acc: 0.7150 - val_mean_squared_error: 0.0029
Epoch 23/50
Epoch 00022: val_loss improved from 0.00286 to 0.00276, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0040 - acc: 0.6998 - mean_squared_error: 0.0040 - val_loss: 0.
0028 - val_acc: 0.7150 - val_mean_squared_error: 0.0028
Epoch 24/50
Epoch 00023: val_loss improved from 0.00276 to 0.00258, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0039 - acc: 0.6986 - mean_squared_error: 0.0039 - val_loss: 0.
0026 - val_acc: 0.7103 - val_mean_squared_error: 0.0026
Epoch 25/50
Epoch 00024: val_loss did not improve
2s - loss: 0.0038 - acc: 0.6980 - mean_squared_error: 0.0038 - val_loss: 0.
0026 - val_acc: 0.7126 - val_mean_squared_error: 0.0026
Epoch 26/50
Epoch 00025: val_loss improved from 0.00258 to 0.00244, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
```

```
3s - loss: 0.0036 - acc: 0.7056 - mean_squared_error: 0.0036 - val_loss: 0.
0024 - val_acc: 0.7056 - val_mean_squared_error: 0.0024
Epoch 27/50
Epoch 00026: val_loss did not improve
2s - loss: 0.0035 - acc: 0.6998 - mean_squared_error: 0.0035 - val_loss: 0.
0025 - val_acc: 0.7126 - val_mean_squared_error: 0.0025
Epoch 28/50
Epoch 00027: val_loss improved from 0.00244 to 0.00238, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0034 - acc: 0.7062 - mean_squared_error: 0.0034 - val_loss: 0.
0024 - val_acc: 0.7103 - val_mean_squared_error: 0.0024
Epoch 29/50
Epoch 00028: val_loss improved from 0.00238 to 0.00227, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0034 - acc: 0.6963 - mean_squared_error: 0.0034 - val_loss: 0.
0023 - val_acc: 0.7173 - val_mean_squared_error: 0.0023
Epoch 30/50
Epoch 00029: val_loss did not improve
2s - loss: 0.0033 - acc: 0.6939 - mean_squared_error: 0.0033 - val_loss: 0.
0023 - val_acc: 0.7196 - val_mean_squared_error: 0.0023
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0032 - acc: 0.7068 - mean_squared_error: 0.0032 - val_loss: 0.
0024 - val_acc: 0.7173 - val_mean_squared_error: 0.0024
Epoch 32/50
Epoch 00031: val_loss improved from 0.00227 to 0.00215, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0032 - acc: 0.7015 - mean_squared_error: 0.0032 - val_loss: 0.
0022 - val_acc: 0.7173 - val_mean_squared_error: 0.0022
Epoch 33/50
Epoch 00032: val_loss did not improve
2s - loss: 0.0031 - acc: 0.7079 - mean_squared_error: 0.0031 - val_loss: 0.
0022 - val_acc: 0.7196 - val_mean_squared_error: 0.0022
Epoch 34/50
Epoch 00033: val_loss improved from 0.00215 to 0.00212, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0030 - acc: 0.7074 - mean_squared_error: 0.0030 - val_loss: 0.
0021 - val_acc: 0.7126 - val_mean_squared_error: 0.0021
Epoch 35/50
Epoch 00034: val_loss improved from 0.00212 to 0.00207, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0030 - acc: 0.7138 - mean_squared_error: 0.0030 - val_loss: 0.
0021 - val_acc: 0.7173 - val_mean_squared_error: 0.0021
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0030 - acc: 0.7120 - mean_squared_error: 0.0030 - val_loss: 0.
0023 - val_acc: 0.7196 - val_mean_squared_error: 0.0023
Epoch 37/50
Epoch 00036: val_loss improved from 0.00207 to 0.00205, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0029 - acc: 0.7074 - mean_squared_error: 0.0029 - val_loss: 0.
0020 - val_acc: 0.7103 - val_mean_squared_error: 0.0020
Epoch 38/50
Epoch 00037: val_loss improved from 0.00205 to 0.00195, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0028 - acc: 0.7150 - mean_squared_error: 0.0028 - val_loss: 0.
0019 - val_acc: 0.7126 - val_mean_squared_error: 0.0019
```

```
Epoch 39/50
Epoch 00038: val_loss improved from 0.00195 to 0.00195, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0028 - acc: 0.7079 - mean_squared_error: 0.0028 - val_loss: 0.
0019 - val_acc: 0.7126 - val_mean_squared_error: 0.0019
Epoch 40/50
Epoch 00039: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7103 - mean_squared_error: 0.0027 - val_loss: 0.
0021 - val_acc: 0.7150 - val_mean_squared_error: 0.0021
Epoch 41/50
Epoch 00040: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7202 - mean_squared_error: 0.0027 - val_loss: 0.
0022 - val_acc: 0.7220 - val_mean_squared_error: 0.0022
Epoch 42/50
Epoch 00041: val_loss improved from 0.00195 to 0.00191, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0026 - acc: 0.7097 - mean_squared_error: 0.0026 - val_loss: 0.
0019 - val_acc: 0.7196 - val_mean_squared_error: 0.0019
Epoch 43/50
Epoch 00042: val_loss improved from 0.00191 to 0.00183, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0026 - acc: 0.7249 - mean_squared_error: 0.0026 - val_loss: 0.
0018 - val_acc: 0.7220 - val_mean_squared_error: 0.0018
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0026 - acc: 0.7161 - mean_squared_error: 0.0026 - val_loss: 0.
0018 - val_acc: 0.7150 - val_mean_squared_error: 0.0018
Epoch 45/50
Epoch 00044: val_loss improved from 0.00183 to 0.00179, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0025 - acc: 0.7261 - mean_squared_error: 0.0025 - val_loss: 0.
0018 - val_acc: 0.7266 - val_mean_squared_error: 0.0018
Epoch 46/50
Epoch 00045: val_loss did not improve
2s - loss: 0.0025 - acc: 0.7266 - mean_squared_error: 0.0025 - val_loss: 0.
0018 - val_acc: 0.7150 - val_mean_squared_error: 0.0018
Epoch 47/50
Epoch 00046: val_loss improved from 0.00179 to 0.00179, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0025 - acc: 0.7167 - mean_squared_error: 0.0025 - val_loss: 0.
0018 - val_acc: 0.7196 - val_mean_squared_error: 0.0018
Epoch 48/50
Epoch 00047: val_loss did not improve
2s - loss: 0.0024 - acc: 0.7109 - mean_squared_error: 0.0024 - val_loss: 0.
0018 - val_acc: 0.7150 - val_mean_squared_error: 0.0018
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0024 - acc: 0.7208 - mean_squared_error: 0.0024 - val_loss: 0.
0020 - val_acc: 0.7360 - val_mean_squared_error: 0.0020
Epoch 50/50
Epoch 00049: val_loss improved from 0.00179 to 0.00175, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0024 - acc: 0.7336 - mean_squared_error: 0.0024 - val_loss: 0.
0017 - val_acc: 0.7243 - val_mean_squared_error: 0.0017
```

```
  Running model: base_model w/opt: Adam
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.00514, saving model to ./mode
l/Adam_model.weights.best.hdf5
4s - loss: 0.0308 - acc: 0.3914 - mean_squared_error: 0.0308 - val_loss: 0.
0051 - val_acc: 0.6963 - val_mean_squared_error: 0.0051
Epoch 2/50
Epoch 00001: val_loss improved from 0.00514 to 0.00498, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0099 - acc: 0.5461 - mean_squared_error: 0.0099 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 3/50
Epoch 00002: val_loss improved from 0.00498 to 0.00356, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0076 - acc: 0.6016 - mean_squared_error: 0.0076 - val_loss: 0.
0036 - val_acc: 0.6939 - val_mean_squared_error: 0.0036
Epoch 4/50
Epoch 00003: val_loss improved from 0.00356 to 0.00323, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0064 - acc: 0.6419 - mean_squared_error: 0.0064 - val_loss: 0.
0032 - val_acc: 0.7009 - val_mean_squared_error: 0.0032
Epoch 5/50
Epoch 00004: val_loss improved from 0.00323 to 0.00282, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0058 - acc: 0.6571 - mean_squared_error: 0.0058 - val_loss: 0.
0028 - val_acc: 0.7173 - val_mean_squared_error: 0.0028
Epoch 6/50
Epoch 00005: val_loss improved from 0.00282 to 0.00241, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0051 - acc: 0.6711 - mean_squared_error: 0.0051 - val_loss: 0.
0024 - val_acc: 0.7150 - val_mean_squared_error: 0.0024
Epoch 7/50
Epoch 00006: val_loss did not improve
2s - loss: 0.0044 - acc: 0.6723 - mean_squared_error: 0.0044 - val_loss: 0.
0033 - val_acc: 0.7360 - val_mean_squared_error: 0.0033
Epoch 8/50
Epoch 00007: val_loss improved from 0.00241 to 0.00203, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0041 - acc: 0.6776 - mean_squared_error: 0.0041 - val_loss: 0.
0020 - val_acc: 0.7173 - val_mean_squared_error: 0.0020
Epoch 9/50
Epoch 00008: val_loss did not improve
2s - loss: 0.0038 - acc: 0.6945 - mean_squared_error: 0.0038 - val_loss: 0.
0021 - val_acc: 0.7196 - val_mean_squared_error: 0.0021
Epoch 10/50
Epoch 00009: val_loss did not improve
2s - loss: 0.0036 - acc: 0.6974 - mean_squared_error: 0.0036 - val_loss: 0.
0023 - val_acc: 0.7173 - val_mean_squared_error: 0.0023
Epoch 11/50
Epoch 00010: val_loss improved from 0.00203 to 0.00184, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0037 - acc: 0.6852 - mean_squared_error: 0.0037 - val_loss: 0.
0018 - val_acc: 0.7103 - val_mean_squared_error: 0.0018
Epoch 12/50
Epoch 00011: val_loss improved from 0.00184 to 0.00172, saving model to ./m
odel/Adam_model.weights.best.hdf5
```

```
3s - loss: 0.0033 - acc: 0.7050 - mean_squared_error: 0.0033 - val_loss: 0.
0017 - val_acc: 0.7477 - val_mean_squared_error: 0.0017
Epoch 13/50
Epoch 00012: val_loss did not improve
2s - loss: 0.0031 - acc: 0.7132 - mean_squared_error: 0.0031 - val_loss: 0.
0019 - val_acc: 0.7430 - val_mean_squared_error: 0.0019
Epoch 14/50
Epoch 00013: val_loss improved from 0.00172 to 0.00166, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0030 - acc: 0.7126 - mean_squared_error: 0.0030 - val_loss: 0.
0017 - val_acc: 0.7453 - val_mean_squared_error: 0.0017
Epoch 15/50
Epoch 00014: val_loss did not improve
2s - loss: 0.0028 - acc: 0.7296 - mean_squared_error: 0.0028 - val_loss: 0.
0022 - val_acc: 0.7407 - val_mean_squared_error: 0.0022
Epoch 16/50
Epoch 00015: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7255 - mean_squared_error: 0.0027 - val_loss: 0.
0018 - val_acc: 0.7079 - val_mean_squared_error: 0.0018
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7208 - mean_squared_error: 0.0027 - val_loss: 0.
0018 - val_acc: 0.7523 - val_mean_squared_error: 0.0018
Epoch 18/50
Epoch 00017: val_loss improved from 0.00166 to 0.00155, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0026 - acc: 0.7190 - mean_squared_error: 0.0026 - val_loss: 0.
0015 - val_acc: 0.7407 - val_mean_squared_error: 0.0015
Epoch 19/50
Epoch 00018: val_loss did not improve
2s - loss: 0.0025 - acc: 0.7354 - mean_squared_error: 0.0025 - val_loss: 0.
0016 - val_acc: 0.7383 - val_mean_squared_error: 0.0016
Epoch 20/50
Epoch 00019: val_loss improved from 0.00155 to 0.00154, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0024 - acc: 0.7477 - mean_squared_error: 0.0024 - val_loss: 0.
0015 - val_acc: 0.7477 - val_mean_squared_error: 0.0015
Epoch 21/50
Epoch 00020: val_loss improved from 0.00154 to 0.00147, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0024 - acc: 0.7395 - mean_squared_error: 0.0024 - val_loss: 0.
0015 - val_acc: 0.7523 - val_mean_squared_error: 0.0015
Epoch 22/50
Epoch 00021: val_loss did not improve
2s - loss: 0.0023 - acc: 0.7313 - mean_squared_error: 0.0023 - val_loss: 0.
0015 - val_acc: 0.7500 - val_mean_squared_error: 0.0015
Epoch 23/50
Epoch 00022: val_loss improved from 0.00147 to 0.00136, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0021 - acc: 0.7477 - mean_squared_error: 0.0021 - val_loss: 0.
0014 - val_acc: 0.7617 - val_mean_squared_error: 0.0014
Epoch 24/50
Epoch 00023: val_loss did not improve
2s - loss: 0.0021 - acc: 0.7377 - mean_squared_error: 0.0021 - val_loss: 0.
0016 - val_acc: 0.7687 - val_mean_squared_error: 0.0016
Epoch 25/50
Epoch 00024: val_loss did not improve
```

```
2s - loss: 0.0022 - acc: 0.7482 - mean_squared_error: 0.0022 - val_loss: 0.
0017 - val_acc: 0.7477 - val_mean_squared_error: 0.0017
Epoch 26/50
Epoch 00025: val_loss did not improve
2s - loss: 0.0021 - acc: 0.7593 - mean_squared_error: 0.0021 - val_loss: 0.
0016 - val_acc: 0.7477 - val_mean_squared_error: 0.0016
Epoch 27/50
Epoch 00026: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7547 - mean_squared_error: 0.0019 - val_loss: 0.
0015 - val_acc: 0.7593 - val_mean_squared_error: 0.0015
Epoch 28/50
Epoch 00027: val_loss did not improve
2s - loss: 0.0020 - acc: 0.7360 - mean_squared_error: 0.0020 - val_loss: 0.
0015 - val_acc: 0.7640 - val_mean_squared_error: 0.0015
Epoch 29/50
Epoch 00028: val_loss improved from 0.00136 to 0.00131, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0019 - acc: 0.7523 - mean_squared_error: 0.0019 - val_loss: 0.
0013 - val_acc: 0.7430 - val_mean_squared_error: 0.0013
Epoch 30/50
Epoch 00029: val_loss improved from 0.00131 to 0.00127, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0018 - acc: 0.7576 - mean_squared_error: 0.0018 - val_loss: 0.
0013 - val_acc: 0.7593 - val_mean_squared_error: 0.0013
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7564 - mean_squared_error: 0.0018 - val_loss: 0.
0017 - val_acc: 0.7523 - val_mean_squared_error: 0.0017
Epoch 32/50
Epoch 00031: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7588 - mean_squared_error: 0.0018 - val_loss: 0.
0013 - val_acc: 0.7570 - val_mean_squared_error: 0.0013
Epoch 33/50
Epoch 00032: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7664 - mean_squared_error: 0.0017 - val_loss: 0.
0013 - val_acc: 0.7500 - val_mean_squared_error: 0.0013
Epoch 34/50
Epoch 00033: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7611 - mean_squared_error: 0.0017 - val_loss: 0.
0014 - val_acc: 0.7547 - val_mean_squared_error: 0.0014
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7739 - mean_squared_error: 0.0016 - val_loss: 0.
0013 - val_acc: 0.7617 - val_mean_squared_error: 0.0013
Epoch 36/50
Epoch 00035: val_loss improved from 0.00127 to 0.00125, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0016 - acc: 0.7821 - mean_squared_error: 0.0016 - val_loss: 0.
0012 - val_acc: 0.7640 - val_mean_squared_error: 0.0012
Epoch 37/50
Epoch 00036: val_loss improved from 0.00125 to 0.00125, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0015 - acc: 0.7734 - mean_squared_error: 0.0015 - val_loss: 0.
0012 - val_acc: 0.7664 - val_mean_squared_error: 0.0012
Epoch 38/50
Epoch 00037: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7716 - mean_squared_error: 0.0015 - val_loss: 0.
```

```
0013 - val_acc: 0.7640 - val_mean_squared_error: 0.0013
Epoch 39/50
Epoch 00038: val_loss improved from 0.00125 to 0.00116, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0015 - acc: 0.7640 - mean_squared_error: 0.0015 - val_loss: 0.
0012 - val_acc: 0.7734 - val_mean_squared_error: 0.0012
Epoch 40/50
Epoch 00039: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7763 - mean_squared_error: 0.0015 - val_loss: 0.
0012 - val_acc: 0.7617 - val_mean_squared_error: 0.0012
Epoch 41/50
Epoch 00040: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7792 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7617 - val_mean_squared_error: 0.0012
Epoch 42/50
Epoch 00041: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7839 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7710 - val_mean_squared_error: 0.0012
Epoch 43/50
Epoch 00042: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7728 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7850 - val_mean_squared_error: 0.0012
Epoch 44/50
Epoch 00043: val_loss improved from 0.00116 to 0.00111, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0014 - acc: 0.7798 - mean_squared_error: 0.0014 - val_loss: 0.
0011 - val_acc: 0.7897 - val_mean_squared_error: 0.0011
Epoch 45/50
Epoch 00044: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7921 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7757 - val_mean_squared_error: 0.0012
Epoch 46/50
Epoch 00045: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7985 - mean_squared_error: 0.0013 - val_loss: 0.
0011 - val_acc: 0.7804 - val_mean_squared_error: 0.0011
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7810 - mean_squared_error: 0.0013 - val_loss: 0.
0012 - val_acc: 0.7687 - val_mean_squared_error: 0.0012
Epoch 48/50
Epoch 00047: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7839 - mean_squared_error: 0.0013 - val_loss: 0.
0012 - val_acc: 0.7734 - val_mean_squared_error: 0.0012
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0012 - acc: 0.7979 - mean_squared_error: 0.0012 - val_loss: 0.
0011 - val_acc: 0.7710 - val_mean_squared_error: 0.0011
Epoch 50/50
Epoch 00049: val_loss improved from 0.00111 to 0.00110, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0012 - acc: 0.8014 - mean_squared_error: 0.0012 - val_loss: 0.
0011 - val_acc: 0.7944 - val_mean_squared_error: 0.0011



  Running model: base_model w/opt: Adamax
Train on 1712 samples, validate on 428 samples
```

```
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.00478, saving model to ./mode
l/Adamax_model.weights.best.hdf5
4s - loss: 0.0280 - acc: 0.4147 - mean_squared_error: 0.0280 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 2/50
Epoch 00001: val_loss did not improve
2s - loss: 0.0106 - acc: 0.5099 - mean_squared_error: 0.0106 - val_loss: 0.
0051 - val_acc: 0.6963 - val_mean_squared_error: 0.0051
Epoch 3/50
Epoch 00002: val_loss improved from 0.00478 to 0.00473, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0091 - acc: 0.5584 - mean_squared_error: 0.0091 - val_loss: 0.
0047 - val_acc: 0.6963 - val_mean_squared_error: 0.0047
Epoch 4/50
Epoch 00003: val_loss improved from 0.00473 to 0.00356, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0081 - acc: 0.6040 - mean_squared_error: 0.0081 - val_loss: 0.
0036 - val_acc: 0.7033 - val_mean_squared_error: 0.0036
Epoch 5/50
Epoch 00004: val_loss improved from 0.00356 to 0.00311, saving model to ./m
odel/Adamax_model.weights.best.hdf5
3s - loss: 0.0071 - acc: 0.6110 - mean_squared_error: 0.0071 - val_loss: 0.
0031 - val_acc: 0.7056 - val_mean_squared_error: 0.0031
Epoch 6/50
Epoch 00005: val_loss improved from 0.00311 to 0.00289, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0065 - acc: 0.6238 - mean_squared_error: 0.0065 - val_loss: 0.
0029 - val_acc: 0.7173 - val_mean_squared_error: 0.0029
Epoch 7/50
Epoch 00006: val_loss did not improve
2s - loss: 0.0061 - acc: 0.6449 - mean_squared_error: 0.0061 - val_loss: 0.
0034 - val_acc: 0.7220 - val_mean_squared_error: 0.0034
Epoch 8/50
Epoch 00007: val_loss improved from 0.00289 to 0.00242, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0057 - acc: 0.6449 - mean_squared_error: 0.0057 - val_loss: 0.
0024 - val_acc: 0.7126 - val_mean_squared_error: 0.0024
Epoch 9/50
Epoch 00008: val_loss did not improve
2s - loss: 0.0055 - acc: 0.6583 - mean_squared_error: 0.0055 - val_loss: 0.
0032 - val_acc: 0.7290 - val_mean_squared_error: 0.0032
Epoch 10/50
Epoch 00009: val_loss improved from 0.00242 to 0.00218, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0051 - acc: 0.6852 - mean_squared_error: 0.0051 - val_loss: 0.
0022 - val_acc: 0.7196 - val_mean_squared_error: 0.0022
Epoch 11/50
Epoch 00010: val_loss did not improve
2s - loss: 0.0047 - acc: 0.6729 - mean_squared_error: 0.0047 - val_loss: 0.
0022 - val_acc: 0.7243 - val_mean_squared_error: 0.0022
Epoch 12/50
Epoch 00011: val_loss did not improve
2s - loss: 0.0046 - acc: 0.6752 - mean_squared_error: 0.0046 - val_loss: 0.
0026 - val_acc: 0.7079 - val_mean_squared_error: 0.0026
Epoch 13/50
Epoch 00012: val_loss improved from 0.00218 to 0.00204, saving model to ./m
```

```
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0044 - acc: 0.6811 - mean_squared_error: 0.0044 - val_loss: 0.
0020 - val_acc: 0.7313 - val_mean_squared_error: 0.0020
Epoch 14/50
Epoch 00013: val_loss did not improve
2s - loss: 0.0043 - acc: 0.7103 - mean_squared_error: 0.0043 - val_loss: 0.
0022 - val_acc: 0.7453 - val_mean_squared_error: 0.0022
Epoch 15/50
Epoch 00014: val_loss improved from 0.00204 to 0.00187, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0039 - acc: 0.7033 - mean_squared_error: 0.0039 - val_loss: 0.
0019 - val_acc: 0.7173 - val_mean_squared_error: 0.0019
Epoch 16/50
Epoch 00015: val_loss did not improve
2s - loss: 0.0040 - acc: 0.6980 - mean_squared_error: 0.0040 - val_loss: 0.
0026 - val_acc: 0.7290 - val_mean_squared_error: 0.0026
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0038 - acc: 0.7021 - mean_squared_error: 0.0038 - val_loss: 0.
0023 - val_acc: 0.7430 - val_mean_squared_error: 0.0023
Epoch 18/50
Epoch 00017: val_loss did not improve
2s - loss: 0.0037 - acc: 0.7167 - mean_squared_error: 0.0037 - val_loss: 0.
0020 - val_acc: 0.7313 - val_mean_squared_error: 0.0020
Epoch 19/50
Epoch 00018: val_loss did not improve
2s - loss: 0.0035 - acc: 0.6939 - mean_squared_error: 0.0035 - val_loss: 0.
0024 - val_acc: 0.7360 - val_mean_squared_error: 0.0024
Epoch 20/50
Epoch 00019: val_loss improved from 0.00187 to 0.00178, saving model to ./m
odel/Adamax_model.weights.best.hdf5
3s - loss: 0.0035 - acc: 0.7050 - mean_squared_error: 0.0035 - val_loss: 0.
0018 - val_acc: 0.7664 - val_mean_squared_error: 0.0018
Epoch 21/50
Epoch 00020: val_loss improved from 0.00178 to 0.00161, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0033 - acc: 0.7091 - mean_squared_error: 0.0033 - val_loss: 0.
0016 - val_acc: 0.7593 - val_mean_squared_error: 0.0016
Epoch 22/50
Epoch 00021: val_loss did not improve
2s - loss: 0.0033 - acc: 0.7161 - mean_squared_error: 0.0033 - val_loss: 0.
0020 - val_acc: 0.7804 - val_mean_squared_error: 0.0020
Epoch 23/50
Epoch 00022: val_loss improved from 0.00161 to 0.00161, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0032 - acc: 0.7085 - mean_squared_error: 0.0032 - val_loss: 0.
0016 - val_acc: 0.7664 - val_mean_squared_error: 0.0016
Epoch 24/50
Epoch 00023: val_loss improved from 0.00161 to 0.00158, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0030 - acc: 0.7243 - mean_squared_error: 0.0030 - val_loss: 0.
0016 - val_acc: 0.7430 - val_mean_squared_error: 0.0016
Epoch 25/50
Epoch 00024: val_loss did not improve
2s - loss: 0.0031 - acc: 0.7284 - mean_squared_error: 0.0031 - val_loss: 0.
0016 - val_acc: 0.7523 - val_mean_squared_error: 0.0016
Epoch 26/50
```

```
Epoch 00025: val_loss improved from 0.00158 to 0.00151, saving model to ./m
odel/Adamax_model.weights.best.hdf5
3s - loss: 0.0030 - acc: 0.7371 - mean_squared_error: 0.0030 - val_loss: 0.
0015 - val_acc: 0.7617 - val_mean_squared_error: 0.0015
Epoch 27/50
Epoch 00026: val_loss did not improve
2s - loss: 0.0029 - acc: 0.7278 - mean_squared_error: 0.0029 - val_loss: 0.
0015 - val_acc: 0.7313 - val_mean_squared_error: 0.0015
Epoch 28/50
Epoch 00027: val_loss did not improve
2s - loss: 0.0028 - acc: 0.7407 - mean_squared_error: 0.0028 - val_loss: 0.
0017 - val_acc: 0.7453 - val_mean_squared_error: 0.0017
Epoch 29/50
Epoch 00028: val_loss improved from 0.00151 to 0.00143, saving model to ./m
odel/Adamax_model.weights.best.hdf5
3s - loss: 0.0027 - acc: 0.7395 - mean_squared_error: 0.0027 - val_loss: 0.
0014 - val_acc: 0.7593 - val_mean_squared_error: 0.0014
Epoch 30/50
Epoch 00029: val_loss did not improve
2s - loss: 0.0025 - acc: 0.7430 - mean_squared_error: 0.0025 - val_loss: 0.
0015 - val_acc: 0.7453 - val_mean_squared_error: 0.0015
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0026 - acc: 0.7465 - mean_squared_error: 0.0026 - val_loss: 0.
0017 - val_acc: 0.7547 - val_mean_squared_error: 0.0017
Epoch 32/50
Epoch 00031: val_loss did not improve
2s - loss: 0.0025 - acc: 0.7477 - mean_squared_error: 0.0025 - val_loss: 0.
0016 - val_acc: 0.7430 - val_mean_squared_error: 0.0016
Epoch 33/50
Epoch 00032: val_loss did not improve
2s - loss: 0.0024 - acc: 0.7453 - mean_squared_error: 0.0024 - val_loss: 0.
0020 - val_acc: 0.7710 - val_mean_squared_error: 0.0020
Epoch 34/50
Epoch 00033: val_loss improved from 0.00143 to 0.00142, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0024 - acc: 0.7629 - mean_squared_error: 0.0024 - val_loss: 0.
0014 - val_acc: 0.7804 - val_mean_squared_error: 0.0014
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0023 - acc: 0.7611 - mean_squared_error: 0.0023 - val_loss: 0.
0015 - val_acc: 0.7687 - val_mean_squared_error: 0.0015
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0022 - acc: 0.7605 - mean_squared_error: 0.0022 - val_loss: 0.
0016 - val_acc: 0.7664 - val_mean_squared_error: 0.0016
Epoch 37/50
Epoch 00036: val_loss improved from 0.00142 to 0.00137, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0021 - acc: 0.7634 - mean_squared_error: 0.0021 - val_loss: 0.
0014 - val_acc: 0.7897 - val_mean_squared_error: 0.0014
Epoch 38/50
Epoch 00037: val_loss did not improve
2s - loss: 0.0021 - acc: 0.7634 - mean_squared_error: 0.0021 - val_loss: 0.
0014 - val_acc: 0.7757 - val_mean_squared_error: 0.0014
Epoch 39/50
Epoch 00038: val_loss improved from 0.00137 to 0.00131, saving model to ./m
```

```
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0021 - acc: 0.7465 - mean_squared_error: 0.0021 - val_loss: 0.
0013 - val_acc: 0.7780 - val_mean_squared_error: 0.0013
Epoch 40/50
Epoch 00039: val_loss did not improve
2s - loss: 0.0020 - acc: 0.7780 - mean_squared_error: 0.0020 - val_loss: 0.
0014 - val_acc: 0.7687 - val_mean_squared_error: 0.0014
Epoch 41/50
Epoch 00040: val_loss did not improve
2s - loss: 0.0020 - acc: 0.7699 - mean_squared_error: 0.0020 - val_loss: 0.
0014 - val_acc: 0.7804 - val_mean_squared_error: 0.0014
Epoch 42/50
Epoch 00041: val_loss improved from 0.00131 to 0.00125, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0019 - acc: 0.7687 - mean_squared_error: 0.0019 - val_loss: 0.
0012 - val_acc: 0.7780 - val_mean_squared_error: 0.0012
Epoch 43/50
Epoch 00042: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7868 - mean_squared_error: 0.0018 - val_loss: 0.
0013 - val_acc: 0.7710 - val_mean_squared_error: 0.0013
Epoch 44/50
Epoch 00043: val_loss improved from 0.00125 to 0.00121, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0018 - acc: 0.7722 - mean_squared_error: 0.0018 - val_loss: 0.
0012 - val_acc: 0.7710 - val_mean_squared_error: 0.0012
Epoch 45/50
Epoch 00044: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7786 - mean_squared_error: 0.0018 - val_loss: 0.
0014 - val_acc: 0.7897 - val_mean_squared_error: 0.0014
Epoch 46/50
Epoch 00045: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7669 - mean_squared_error: 0.0017 - val_loss: 0.
0013 - val_acc: 0.7827 - val_mean_squared_error: 0.0013
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7850 - mean_squared_error: 0.0017 - val_loss: 0.
0013 - val_acc: 0.7757 - val_mean_squared_error: 0.0013
Epoch 48/50
Epoch 00047: val_loss improved from 0.00121 to 0.00120, saving model to ./m
odel/Adamax_model.weights.best.hdf5
3s - loss: 0.0016 - acc: 0.7786 - mean_squared_error: 0.0016 - val_loss: 0.
0012 - val_acc: 0.7921 - val_mean_squared_error: 0.0012
Epoch 49/50
Epoch 00048: val_loss improved from 0.00120 to 0.00118, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0016 - acc: 0.7798 - mean_squared_error: 0.0016 - val_loss: 0.
0012 - val_acc: 0.7874 - val_mean_squared_error: 0.0012
Epoch 50/50
Epoch 00049: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7728 - mean_squared_error: 0.0016 - val_loss: 0.
0013 - val_acc: 0.7710 - val_mean_squared_error: 0.0013



  Running model: base_model w/opt: Nadam
Train on 1712 samples, validate on 428 samples
Epoch 1/50
```

```
Epoch 00000: val_loss improved from inf to 0.00644, saving model to ./mode
l/Nadam_model.weights.best.hdf5
4s - loss: 0.0556 - acc: 0.3727 - mean_squared_error: 0.0556 - val_loss: 0.
0064 - val_acc: 0.6963 - val_mean_squared_error: 0.0064
Epoch 2/50
Epoch 00001: val_loss did not improve
2s - loss: 0.0119 - acc: 0.5105 - mean_squared_error: 0.0119 - val_loss: 0.
0072 - val_acc: 0.6963 - val_mean_squared_error: 0.0072
Epoch 3/50
Epoch 00002: val_loss improved from 0.00644 to 0.00389, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0094 - acc: 0.5613 - mean_squared_error: 0.0094 - val_loss: 0.
0039 - val_acc: 0.6963 - val_mean_squared_error: 0.0039
Epoch 4/50
Epoch 00003: val_loss did not improve
2s - loss: 0.0080 - acc: 0.5929 - mean_squared_error: 0.0080 - val_loss: 0.
0061 - val_acc: 0.6963 - val_mean_squared_error: 0.0061
Epoch 5/50
Epoch 00004: val_loss did not improve
2s - loss: 0.0070 - acc: 0.6168 - mean_squared_error: 0.0070 - val_loss: 0.
0044 - val_acc: 0.6963 - val_mean_squared_error: 0.0044
Epoch 6/50
Epoch 00005: val_loss improved from 0.00389 to 0.00328, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0062 - acc: 0.6379 - mean_squared_error: 0.0062 - val_loss: 0.
0033 - val_acc: 0.6963 - val_mean_squared_error: 0.0033
Epoch 7/50
Epoch 00006: val_loss did not improve
2s - loss: 0.0060 - acc: 0.6501 - mean_squared_error: 0.0060 - val_loss: 0.
0036 - val_acc: 0.6963 - val_mean_squared_error: 0.0036
Epoch 8/50
Epoch 00007: val_loss improved from 0.00328 to 0.00301, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0051 - acc: 0.6735 - mean_squared_error: 0.0051 - val_loss: 0.
0030 - val_acc: 0.7033 - val_mean_squared_error: 0.0030
Epoch 9/50
Epoch 00008: val_loss did not improve
2s - loss: 0.0046 - acc: 0.6676 - mean_squared_error: 0.0046 - val_loss: 0.
0032 - val_acc: 0.7056 - val_mean_squared_error: 0.0032
Epoch 10/50
Epoch 00009: val_loss improved from 0.00301 to 0.00225, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0042 - acc: 0.6776 - mean_squared_error: 0.0042 - val_loss: 0.
0023 - val_acc: 0.7500 - val_mean_squared_error: 0.0023
Epoch 11/50
Epoch 00010: val_loss improved from 0.00225 to 0.00193, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0039 - acc: 0.6963 - mean_squared_error: 0.0039 - val_loss: 0.
0019 - val_acc: 0.7593 - val_mean_squared_error: 0.0019
Epoch 12/50
Epoch 00011: val_loss did not improve
2s - loss: 0.0036 - acc: 0.6881 - mean_squared_error: 0.0036 - val_loss: 0.
0021 - val_acc: 0.7079 - val_mean_squared_error: 0.0021
Epoch 13/50
Epoch 00012: val_loss improved from 0.00193 to 0.00177, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0033 - acc: 0.7161 - mean_squared_error: 0.0033 - val_loss: 0.
```

```
0018 - val_acc: 0.7547 - val_mean_squared_error: 0.0018
Epoch 14/50
Epoch 00013: val_loss did not improve
2s - loss: 0.0030 - acc: 0.7161 - mean_squared_error: 0.0030 - val_loss: 0.
0020 - val_acc: 0.7290 - val_mean_squared_error: 0.0020
Epoch 15/50
Epoch 00014: val_loss did not improve
2s - loss: 0.0029 - acc: 0.7074 - mean_squared_error: 0.0029 - val_loss: 0.
0018 - val_acc: 0.7523 - val_mean_squared_error: 0.0018
Epoch 16/50
Epoch 00015: val_loss improved from 0.00177 to 0.00156, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0028 - acc: 0.7173 - mean_squared_error: 0.0028 - val_loss: 0.
0016 - val_acc: 0.7640 - val_mean_squared_error: 0.0016
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0026 - acc: 0.7272 - mean_squared_error: 0.0026 - val_loss: 0.
0019 - val_acc: 0.7290 - val_mean_squared_error: 0.0019
Epoch 18/50
Epoch 00017: val_loss improved from 0.00156 to 0.00156, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0024 - acc: 0.7354 - mean_squared_error: 0.0024 - val_loss: 0.
0016 - val_acc: 0.7734 - val_mean_squared_error: 0.0016
Epoch 19/50
Epoch 00018: val_loss did not improve
2s - loss: 0.0023 - acc: 0.7307 - mean_squared_error: 0.0023 - val_loss: 0.
0019 - val_acc: 0.7804 - val_mean_squared_error: 0.0019
Epoch 20/50
Epoch 00019: val_loss improved from 0.00156 to 0.00152, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0023 - acc: 0.7377 - mean_squared_error: 0.0023 - val_loss: 0.
0015 - val_acc: 0.7593 - val_mean_squared_error: 0.0015
Epoch 21/50
Epoch 00020: val_loss did not improve
2s - loss: 0.0022 - acc: 0.7366 - mean_squared_error: 0.0022 - val_loss: 0.
0017 - val_acc: 0.7360 - val_mean_squared_error: 0.0017
Epoch 22/50
Epoch 00021: val_loss did not improve
2s - loss: 0.0021 - acc: 0.7401 - mean_squared_error: 0.0021 - val_loss: 0.
0018 - val_acc: 0.7687 - val_mean_squared_error: 0.0018
Epoch 23/50
Epoch 00022: val_loss improved from 0.00152 to 0.00141, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0020 - acc: 0.7482 - mean_squared_error: 0.0020 - val_loss: 0.
0014 - val_acc: 0.7664 - val_mean_squared_error: 0.0014
Epoch 24/50
Epoch 00023: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7482 - mean_squared_error: 0.0019 - val_loss: 0.
0015 - val_acc: 0.7430 - val_mean_squared_error: 0.0015
Epoch 25/50
Epoch 00024: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7564 - mean_squared_error: 0.0019 - val_loss: 0.
0015 - val_acc: 0.7710 - val_mean_squared_error: 0.0015
Epoch 26/50
Epoch 00025: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7629 - mean_squared_error: 0.0018 - val_loss: 0.
0015 - val_acc: 0.7570 - val_mean_squared_error: 0.0015
```

```
Epoch 27/50
Epoch 00026: val_loss improved from 0.00141 to 0.00123, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0018 - acc: 0.7675 - mean_squared_error: 0.0018 - val_loss: 0.
0012 - val_acc: 0.7850 - val_mean_squared_error: 0.0012
Epoch 28/50
Epoch 00027: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7629 - mean_squared_error: 0.0017 - val_loss: 0.
0014 - val_acc: 0.7921 - val_mean_squared_error: 0.0014
Epoch 29/50
Epoch 00028: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7488 - mean_squared_error: 0.0016 - val_loss: 0.
0013 - val_acc: 0.8061 - val_mean_squared_error: 0.0013
Epoch 30/50
Epoch 00029: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7850 - mean_squared_error: 0.0016 - val_loss: 0.
0013 - val_acc: 0.7874 - val_mean_squared_error: 0.0013
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7769 - mean_squared_error: 0.0015 - val_loss: 0.
0013 - val_acc: 0.7827 - val_mean_squared_error: 0.0013
Epoch 32/50
Epoch 00031: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7757 - mean_squared_error: 0.0015 - val_loss: 0.
0017 - val_acc: 0.7897 - val_mean_squared_error: 0.0017
Epoch 33/50
Epoch 00032: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7780 - mean_squared_error: 0.0015 - val_loss: 0.
0014 - val_acc: 0.7757 - val_mean_squared_error: 0.0014
Epoch 34/50
Epoch 00033: val_loss improved from 0.00123 to 0.00122, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0014 - acc: 0.7827 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.8061 - val_mean_squared_error: 0.0012
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7792 - mean_squared_error: 0.0014 - val_loss: 0.
0013 - val_acc: 0.8014 - val_mean_squared_error: 0.0013
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7868 - mean_squared_error: 0.0014 - val_loss: 0.
0013 - val_acc: 0.7687 - val_mean_squared_error: 0.0013
Epoch 37/50
Epoch 00036: val_loss improved from 0.00122 to 0.00114, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0013 - acc: 0.7886 - mean_squared_error: 0.0013 - val_loss: 0.
0011 - val_acc: 0.8014 - val_mean_squared_error: 0.0011
Epoch 38/50
Epoch 00037: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7979 - mean_squared_error: 0.0013 - val_loss: 0.
0012 - val_acc: 0.7967 - val_mean_squared_error: 0.0012
Epoch 39/50
Epoch 00038: val_loss did not improve
2s - loss: 0.0012 - acc: 0.7973 - mean_squared_error: 0.0012 - val_loss: 0.
0014 - val_acc: 0.7944 - val_mean_squared_error: 0.0014
Epoch 40/50
Epoch 00039: val_loss did not improve
```

```
2s - loss: 0.0012 - acc: 0.8049 - mean_squared_error: 0.0012 - val_loss: 0.
0012 - val_acc: 0.7897 - val_mean_squared_error: 0.0012
Epoch 41/50
Epoch 00040: val_loss did not improve
2s - loss: 0.0012 - acc: 0.8014 - mean_squared_error: 0.0012 - val_loss: 0.
0013 - val_acc: 0.7921 - val_mean_squared_error: 0.0013
Epoch 42/50
Epoch 00041: val_loss improved from 0.00114 to 0.00112, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0012 - acc: 0.8067 - mean_squared_error: 0.0012 - val_loss: 0.
0011 - val_acc: 0.7897 - val_mean_squared_error: 0.0011
Epoch 43/50
Epoch 00042: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8084 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.8061 - val_mean_squared_error: 0.0011
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8148 - mean_squared_error: 0.0011 - val_loss: 0.
0012 - val_acc: 0.7991 - val_mean_squared_error: 0.0012
Epoch 45/50
Epoch 00044: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8107 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.8061 - val_mean_squared_error: 0.0011
Epoch 46/50
Epoch 00045: val_loss improved from 0.00112 to 0.00108, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0011 - acc: 0.8102 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.8014 - val_mean_squared_error: 0.0011
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8078 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.7967 - val_mean_squared_error: 0.0011
Epoch 48/50
Epoch 00047: val_loss improved from 0.00108 to 0.00108, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0011 - acc: 0.8131 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.7991 - val_mean_squared_error: 0.0011
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0010 - acc: 0.8189 - mean_squared_error: 0.0010 - val_loss: 0.
0012 - val_acc: 0.8107 - val_mean_squared_error: 0.0012
Epoch 50/50
Epoch 00049: val_loss improved from 0.00108 to 0.00107, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 9.9799e-04 - acc: 0.8096 - mean_squared_error: 9.9799e-04 - val_
loss: 0.0011 - val_acc: 0.8084 - val_mean_squared_error: 0.0011



  Running model: bigger_base_model w/opt: SGD
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.01324, saving model to ./mode
l/SGD_model.weights.best.hdf5
3s - loss: 0.0673 - acc: 0.2623 - mean_squared_error: 0.0673 - val_loss: 0.
0132 - val_acc: 0.6963 - val_mean_squared_error: 0.0132
Epoch 2/50
```

```
Epoch 00001: val_loss improved from 0.01324 to 0.01158, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0312 - acc: 0.3908 - mean_squared_error: 0.0312 - val_loss: 0.
0116 - val_acc: 0.6963 - val_mean_squared_error: 0.0116
Epoch 3/50
Epoch 00002: val_loss improved from 0.01158 to 0.00999, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0274 - acc: 0.4095 - mean_squared_error: 0.0274 - val_loss: 0.
0100 - val_acc: 0.6963 - val_mean_squared_error: 0.0100
Epoch 4/50
Epoch 00003: val_loss improved from 0.00999 to 0.00948, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0249 - acc: 0.4048 - mean_squared_error: 0.0249 - val_loss: 0.
0095 - val_acc: 0.6963 - val_mean_squared_error: 0.0095
Epoch 5/50
Epoch 00004: val_loss improved from 0.00948 to 0.00925, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0233 - acc: 0.4311 - mean_squared_error: 0.0233 - val_loss: 0.
0093 - val_acc: 0.6963 - val_mean_squared_error: 0.0093
Epoch 6/50
Epoch 00005: val_loss improved from 0.00925 to 0.00860, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0216 - acc: 0.4433 - mean_squared_error: 0.0216 - val_loss: 0.
0086 - val_acc: 0.6963 - val_mean_squared_error: 0.0086
Epoch 7/50
Epoch 00006: val_loss improved from 0.00860 to 0.00771, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0205 - acc: 0.4515 - mean_squared_error: 0.0205 - val_loss: 0.
0077 - val_acc: 0.6963 - val_mean_squared_error: 0.0077
Epoch 8/50
Epoch 00007: val_loss improved from 0.00771 to 0.00760, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0195 - acc: 0.4451 - mean_squared_error: 0.0195 - val_loss: 0.
0076 - val_acc: 0.6963 - val_mean_squared_error: 0.0076
Epoch 9/50
Epoch 00008: val_loss improved from 0.00760 to 0.00752, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0186 - acc: 0.4790 - mean_squared_error: 0.0186 - val_loss: 0.
0075 - val_acc: 0.6963 - val_mean_squared_error: 0.0075
Epoch 10/50
Epoch 00009: val_loss improved from 0.00752 to 0.00685, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0179 - acc: 0.4737 - mean_squared_error: 0.0179 - val_loss: 0.
0069 - val_acc: 0.6963 - val_mean_squared_error: 0.0069
Epoch 11/50
Epoch 00010: val_loss did not improve
2s - loss: 0.0169 - acc: 0.4626 - mean_squared_error: 0.0169 - val_loss: 0.
0069 - val_acc: 0.6963 - val_mean_squared_error: 0.0069
Epoch 12/50
Epoch 00011: val_loss improved from 0.00685 to 0.00632, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0163 - acc: 0.4790 - mean_squared_error: 0.0163 - val_loss: 0.
0063 - val_acc: 0.6963 - val_mean_squared_error: 0.0063
Epoch 13/50
Epoch 00012: val_loss improved from 0.00632 to 0.00623, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0159 - acc: 0.4749 - mean_squared_error: 0.0159 - val_loss: 0.
```

```
0062 - val_acc: 0.6963 - val_mean_squared_error: 0.0062
Epoch 14/50
Epoch 00013: val_loss improved from 0.00623 to 0.00607, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0152 - acc: 0.4778 - mean_squared_error: 0.0152 - val_loss: 0.
0061 - val_acc: 0.6963 - val_mean_squared_error: 0.0061
Epoch 15/50
Epoch 00014: val_loss improved from 0.00607 to 0.00605, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0151 - acc: 0.4755 - mean_squared_error: 0.0151 - val_loss: 0.
0061 - val_acc: 0.6963 - val_mean_squared_error: 0.0061
Epoch 16/50
Epoch 00015: val_loss improved from 0.00605 to 0.00584, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0144 - acc: 0.4895 - mean_squared_error: 0.0144 - val_loss: 0.
0058 - val_acc: 0.6963 - val_mean_squared_error: 0.0058
Epoch 17/50
Epoch 00016: val_loss improved from 0.00584 to 0.00548, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0141 - acc: 0.5134 - mean_squared_error: 0.0141 - val_loss: 0.
0055 - val_acc: 0.6963 - val_mean_squared_error: 0.0055
Epoch 18/50
Epoch 00017: val_loss improved from 0.00548 to 0.00535, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0138 - acc: 0.4901 - mean_squared_error: 0.0138 - val_loss: 0.
0054 - val_acc: 0.6963 - val_mean_squared_error: 0.0054
Epoch 19/50
Epoch 00018: val_loss improved from 0.00535 to 0.00533, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0137 - acc: 0.4988 - mean_squared_error: 0.0137 - val_loss: 0.
0053 - val_acc: 0.6963 - val_mean_squared_error: 0.0053
Epoch 20/50
Epoch 00019: val_loss did not improve
2s - loss: 0.0136 - acc: 0.5018 - mean_squared_error: 0.0136 - val_loss: 0.
0053 - val_acc: 0.6963 - val_mean_squared_error: 0.0053
Epoch 21/50
Epoch 00020: val_loss did not improve
2s - loss: 0.0133 - acc: 0.5123 - mean_squared_error: 0.0133 - val_loss: 0.
0054 - val_acc: 0.6963 - val_mean_squared_error: 0.0054
Epoch 22/50
Epoch 00021: val_loss improved from 0.00533 to 0.00528, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0130 - acc: 0.4918 - mean_squared_error: 0.0130 - val_loss: 0.
0053 - val_acc: 0.6963 - val_mean_squared_error: 0.0053
Epoch 23/50
Epoch 00022: val_loss improved from 0.00528 to 0.00516, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0129 - acc: 0.5035 - mean_squared_error: 0.0129 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 24/50
Epoch 00023: val_loss did not improve
2s - loss: 0.0126 - acc: 0.5123 - mean_squared_error: 0.0126 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 25/50
Epoch 00024: val_loss did not improve
2s - loss: 0.0125 - acc: 0.5129 - mean_squared_error: 0.0125 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
```

```
Epoch 26/50
Epoch 00025: val_loss did not improve
2s - loss: 0.0123 - acc: 0.5269 - mean_squared_error: 0.0123 - val_loss: 0.
0056 - val_acc: 0.6963 - val_mean_squared_error: 0.0056
Epoch 27/50
Epoch 00026: val_loss did not improve
2s - loss: 0.0123 - acc: 0.5134 - mean_squared_error: 0.0123 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 28/50
Epoch 00027: val_loss did not improve
2s - loss: 0.0120 - acc: 0.5251 - mean_squared_error: 0.0120 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 29/50
Epoch 00028: val_loss did not improve
2s - loss: 0.0120 - acc: 0.5368 - mean_squared_error: 0.0120 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 30/50
Epoch 00029: val_loss improved from 0.00516 to 0.00508, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0118 - acc: 0.5345 - mean_squared_error: 0.0118 - val_loss: 0.
0051 - val_acc: 0.6963 - val_mean_squared_error: 0.0051
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0118 - acc: 0.5082 - mean_squared_error: 0.0118 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 32/50
Epoch 00031: val_loss improved from 0.00508 to 0.00507, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0117 - acc: 0.5169 - mean_squared_error: 0.0117 - val_loss: 0.
0051 - val_acc: 0.6963 - val_mean_squared_error: 0.0051
Epoch 33/50
Epoch 00032: val_loss improved from 0.00507 to 0.00500, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0116 - acc: 0.5532 - mean_squared_error: 0.0116 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 34/50
Epoch 00033: val_loss improved from 0.00500 to 0.00484, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0115 - acc: 0.5450 - mean_squared_error: 0.0115 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0114 - acc: 0.5204 - mean_squared_error: 0.0114 - val_loss: 0.
0053 - val_acc: 0.6963 - val_mean_squared_error: 0.0053
Epoch 36/50
Epoch 00035: val_loss improved from 0.00484 to 0.00477, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0114 - acc: 0.5549 - mean_squared_error: 0.0114 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 37/50
Epoch 00036: val_loss did not improve
2s - loss: 0.0112 - acc: 0.5520 - mean_squared_error: 0.0112 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 38/50
Epoch 00037: val_loss did not improve
2s - loss: 0.0111 - acc: 0.5426 - mean_squared_error: 0.0111 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
```

```
Epoch 39/50
Epoch 00038: val_loss did not improve
2s - loss: 0.0109 - acc: 0.5403 - mean_squared_error: 0.0109 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 40/50
Epoch 00039: val_loss did not improve
2s - loss: 0.0110 - acc: 0.5561 - mean_squared_error: 0.0110 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 41/50
Epoch 00040: val_loss did not improve
2s - loss: 0.0108 - acc: 0.5403 - mean_squared_error: 0.0108 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 42/50
Epoch 00041: val_loss did not improve
2s - loss: 0.0109 - acc: 0.5432 - mean_squared_error: 0.0109 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 43/50
Epoch 00042: val_loss did not improve
2s - loss: 0.0107 - acc: 0.5088 - mean_squared_error: 0.0107 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0105 - acc: 0.5602 - mean_squared_error: 0.0105 - val_loss: 0.
0053 - val_acc: 0.6963 - val_mean_squared_error: 0.0053
Epoch 45/50
Epoch 00044: val_loss improved from 0.00477 to 0.00475, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0107 - acc: 0.5380 - mean_squared_error: 0.0107 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 46/50
Epoch 00045: val_loss did not improve
2s - loss: 0.0105 - acc: 0.5724 - mean_squared_error: 0.0105 - val_loss: 0.
0051 - val_acc: 0.6963 - val_mean_squared_error: 0.0051
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0104 - acc: 0.5561 - mean_squared_error: 0.0104 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 48/50
Epoch 00047: val_loss improved from 0.00475 to 0.00473, saving model to ./m
odel/SGD_model.weights.best.hdf5
2s - loss: 0.0103 - acc: 0.5689 - mean_squared_error: 0.0103 - val_loss: 0.
0047 - val_acc: 0.6963 - val_mean_squared_error: 0.0047
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0104 - acc: 0.5520 - mean_squared_error: 0.0104 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 50/50
Epoch 00049: val_loss did not improve
2s - loss: 0.0102 - acc: 0.5730 - mean_squared_error: 0.0102 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049



  Running model: bigger_base_model w/opt: RMSprop
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.00759, saving model to ./mode
```

```
l/RMSprop_model.weights.best.hdf5
4s - loss: 0.2430 - acc: 0.3984 - mean_squared_error: 0.2430 - val_loss: 0.
0076 - val_acc: 0.6963 - val_mean_squared_error: 0.0076
Epoch 2/50
Epoch 00001: val_loss did not improve
2s - loss: 0.0196 - acc: 0.4930 - mean_squared_error: 0.0196 - val_loss: 0.
0104 - val_acc: 0.6963 - val_mean_squared_error: 0.0104
Epoch 3/50
Epoch 00002: val_loss improved from 0.00759 to 0.00448, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0128 - acc: 0.5537 - mean_squared_error: 0.0128 - val_loss: 0.
0045 - val_acc: 0.6963 - val_mean_squared_error: 0.0045
Epoch 4/50
Epoch 00003: val_loss improved from 0.00448 to 0.00414, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0086 - acc: 0.6133 - mean_squared_error: 0.0086 - val_loss: 0.
0041 - val_acc: 0.6963 - val_mean_squared_error: 0.0041
Epoch 5/50
Epoch 00004: val_loss did not improve
2s - loss: 0.0072 - acc: 0.6507 - mean_squared_error: 0.0072 - val_loss: 0.
0064 - val_acc: 0.6963 - val_mean_squared_error: 0.0064
Epoch 6/50
Epoch 00005: val_loss did not improve
2s - loss: 0.0064 - acc: 0.6706 - mean_squared_error: 0.0064 - val_loss: 0.
0068 - val_acc: 0.7056 - val_mean_squared_error: 0.0068
Epoch 7/50
Epoch 00006: val_loss improved from 0.00414 to 0.00382, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0056 - acc: 0.6793 - mean_squared_error: 0.0056 - val_loss: 0.
0038 - val_acc: 0.7033 - val_mean_squared_error: 0.0038
Epoch 8/50
Epoch 00007: val_loss did not improve
2s - loss: 0.0049 - acc: 0.6922 - mean_squared_error: 0.0049 - val_loss: 0.
0040 - val_acc: 0.7103 - val_mean_squared_error: 0.0040
Epoch 9/50
Epoch 00008: val_loss improved from 0.00382 to 0.00322, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0045 - acc: 0.6857 - mean_squared_error: 0.0045 - val_loss: 0.
0032 - val_acc: 0.7009 - val_mean_squared_error: 0.0032
Epoch 10/50
Epoch 00009: val_loss improved from 0.00322 to 0.00243, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0043 - acc: 0.7085 - mean_squared_error: 0.0043 - val_loss: 0.
0024 - val_acc: 0.7243 - val_mean_squared_error: 0.0024
Epoch 11/50
Epoch 00010: val_loss improved from 0.00243 to 0.00232, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0039 - acc: 0.7150 - mean_squared_error: 0.0039 - val_loss: 0.
0023 - val_acc: 0.7103 - val_mean_squared_error: 0.0023
Epoch 12/50
Epoch 00011: val_loss did not improve
2s - loss: 0.0036 - acc: 0.7179 - mean_squared_error: 0.0036 - val_loss: 0.
0026 - val_acc: 0.7150 - val_mean_squared_error: 0.0026
Epoch 13/50
Epoch 00012: val_loss improved from 0.00232 to 0.00197, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0033 - acc: 0.7155 - mean_squared_error: 0.0033 - val_loss: 0.
```

```
0020 - val_acc: 0.7079 - val_mean_squared_error: 0.0020
Epoch 14/50
Epoch 00013: val_loss did not improve
2s - loss: 0.0030 - acc: 0.7167 - mean_squared_error: 0.0030 - val_loss: 0.
0022 - val_acc: 0.7079 - val_mean_squared_error: 0.0022
Epoch 15/50
Epoch 00014: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7202 - mean_squared_error: 0.0027 - val_loss: 0.
0037 - val_acc: 0.7336 - val_mean_squared_error: 0.0037
Epoch 16/50
Epoch 00015: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7249 - mean_squared_error: 0.0027 - val_loss: 0.
0021 - val_acc: 0.7383 - val_mean_squared_error: 0.0021
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0024 - acc: 0.7290 - mean_squared_error: 0.0024 - val_loss: 0.
0024 - val_acc: 0.7150 - val_mean_squared_error: 0.0024
Epoch 18/50
Epoch 00017: val_loss did not improve
2s - loss: 0.0024 - acc: 0.7407 - mean_squared_error: 0.0024 - val_loss: 0.
0024 - val_acc: 0.7150 - val_mean_squared_error: 0.0024
Epoch 19/50
Epoch 00018: val_loss improved from 0.00197 to 0.00165, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0023 - acc: 0.7471 - mean_squared_error: 0.0023 - val_loss: 0.
0017 - val_acc: 0.7336 - val_mean_squared_error: 0.0017
Epoch 20/50
Epoch 00019: val_loss did not improve
2s - loss: 0.0022 - acc: 0.7389 - mean_squared_error: 0.0022 - val_loss: 0.
0017 - val_acc: 0.7336 - val_mean_squared_error: 0.0017
Epoch 21/50
Epoch 00020: val_loss improved from 0.00165 to 0.00151, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0021 - acc: 0.7506 - mean_squared_error: 0.0021 - val_loss: 0.
0015 - val_acc: 0.7500 - val_mean_squared_error: 0.0015
Epoch 22/50
Epoch 00021: val_loss did not improve
2s - loss: 0.0021 - acc: 0.7599 - mean_squared_error: 0.0021 - val_loss: 0.
0016 - val_acc: 0.7687 - val_mean_squared_error: 0.0016
Epoch 23/50
Epoch 00022: val_loss did not improve
2s - loss: 0.0020 - acc: 0.7518 - mean_squared_error: 0.0020 - val_loss: 0.
0016 - val_acc: 0.7500 - val_mean_squared_error: 0.0016
Epoch 24/50
Epoch 00023: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7453 - mean_squared_error: 0.0019 - val_loss: 0.
0018 - val_acc: 0.7477 - val_mean_squared_error: 0.0018
Epoch 25/50
Epoch 00024: val_loss improved from 0.00151 to 0.00135, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0019 - acc: 0.7664 - mean_squared_error: 0.0019 - val_loss: 0.
0014 - val_acc: 0.7687 - val_mean_squared_error: 0.0014
Epoch 26/50
Epoch 00025: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7640 - mean_squared_error: 0.0019 - val_loss: 0.
0023 - val_acc: 0.7640 - val_mean_squared_error: 0.0023
Epoch 27/50
```

```
Epoch 00026: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7576 - mean_squared_error: 0.0018 - val_loss: 0.
0015 - val_acc: 0.7593 - val_mean_squared_error: 0.0015
Epoch 28/50
Epoch 00027: val_loss improved from 0.00135 to 0.00133, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0017 - acc: 0.7669 - mean_squared_error: 0.0017 - val_loss: 0.
0013 - val_acc: 0.7710 - val_mean_squared_error: 0.0013
Epoch 29/50
Epoch 00028: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7693 - mean_squared_error: 0.0017 - val_loss: 0.
0014 - val_acc: 0.7593 - val_mean_squared_error: 0.0014
Epoch 30/50
Epoch 00029: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7728 - mean_squared_error: 0.0017 - val_loss: 0.
0014 - val_acc: 0.7640 - val_mean_squared_error: 0.0014
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7693 - mean_squared_error: 0.0016 - val_loss: 0.
0020 - val_acc: 0.7407 - val_mean_squared_error: 0.0020
Epoch 32/50
Epoch 00031: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7722 - mean_squared_error: 0.0015 - val_loss: 0.
0014 - val_acc: 0.7710 - val_mean_squared_error: 0.0014
Epoch 33/50
Epoch 00032: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7810 - mean_squared_error: 0.0015 - val_loss: 0.
0014 - val_acc: 0.7827 - val_mean_squared_error: 0.0014
Epoch 34/50
Epoch 00033: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7815 - mean_squared_error: 0.0015 - val_loss: 0.
0015 - val_acc: 0.8037 - val_mean_squared_error: 0.0015
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7862 - mean_squared_error: 0.0014 - val_loss: 0.
0013 - val_acc: 0.7944 - val_mean_squared_error: 0.0013
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7921 - mean_squared_error: 0.0014 - val_loss: 0.
0015 - val_acc: 0.7757 - val_mean_squared_error: 0.0015
Epoch 37/50
Epoch 00036: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7810 - mean_squared_error: 0.0014 - val_loss: 0.
0018 - val_acc: 0.7967 - val_mean_squared_error: 0.0018
Epoch 38/50
Epoch 00037: val_loss improved from 0.00133 to 0.00130, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0013 - acc: 0.7891 - mean_squared_error: 0.0013 - val_loss: 0.
0013 - val_acc: 0.7804 - val_mean_squared_error: 0.0013
Epoch 39/50
Epoch 00038: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7909 - mean_squared_error: 0.0013 - val_loss: 0.
0014 - val_acc: 0.7687 - val_mean_squared_error: 0.0014
Epoch 40/50
Epoch 00039: val_loss improved from 0.00130 to 0.00123, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0013 - acc: 0.7950 - mean_squared_error: 0.0013 - val_loss: 0.
```

```
0012 - val_acc: 0.7827 - val_mean_squared_error: 0.0012
Epoch 41/50
Epoch 00040: val_loss did not improve
2s - loss: 0.0012 - acc: 0.8026 - mean_squared_error: 0.0012 - val_loss: 0.
0018 - val_acc: 0.7640 - val_mean_squared_error: 0.0018
Epoch 42/50
Epoch 00041: val_loss improved from 0.00123 to 0.00121, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0012 - acc: 0.8020 - mean_squared_error: 0.0012 - val_loss: 0.
0012 - val_acc: 0.7640 - val_mean_squared_error: 0.0012
Epoch 43/50
Epoch 00042: val_loss did not improve
2s - loss: 0.0012 - acc: 0.7985 - mean_squared_error: 0.0012 - val_loss: 0.
0012 - val_acc: 0.7617 - val_mean_squared_error: 0.0012
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8078 - mean_squared_error: 0.0011 - val_loss: 0.
0013 - val_acc: 0.7757 - val_mean_squared_error: 0.0013
Epoch 45/50
Epoch 00044: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8037 - mean_squared_error: 0.0011 - val_loss: 0.
0013 - val_acc: 0.7757 - val_mean_squared_error: 0.0013
Epoch 46/50
Epoch 00045: val_loss improved from 0.00121 to 0.00110, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
2s - loss: 0.0011 - acc: 0.8032 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.7944 - val_mean_squared_error: 0.0011
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0011 - acc: 0.7926 - mean_squared_error: 0.0011 - val_loss: 0.
0012 - val_acc: 0.7687 - val_mean_squared_error: 0.0012
Epoch 48/50
Epoch 00047: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8131 - mean_squared_error: 0.0011 - val_loss: 0.
0015 - val_acc: 0.7897 - val_mean_squared_error: 0.0015
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0010 - acc: 0.8183 - mean_squared_error: 0.0010 - val_loss: 0.
0012 - val_acc: 0.7780 - val_mean_squared_error: 0.0012
Epoch 50/50
Epoch 00049: val_loss did not improve
2s - loss: 0.0010 - acc: 0.8072 - mean_squared_error: 0.0010 - val_loss: 0.
0012 - val_acc: 0.7757 - val_mean_squared_error: 0.0012



  Running model: bigger_base_model w/opt: Adagrad
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.01305, saving model to ./mode
l/Adagrad_model.weights.best.hdf5
3s - loss: 8.8295 - acc: 0.3651 - mean_squared_error: 8.8295 - val_loss: 0.
0130 - val_acc: 0.6963 - val_mean_squared_error: 0.0130
Epoch 2/50
Epoch 00001: val_loss improved from 0.01305 to 0.00495, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0218 - acc: 0.4720 - mean_squared_error: 0.0218 - val_loss: 0.
```

```
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 3/50
Epoch 00002: val_loss did not improve
2s - loss: 0.0196 - acc: 0.5450 - mean_squared_error: 0.0196 - val_loss: 0.
0101 - val_acc: 0.6963 - val_mean_squared_error: 0.0101
Epoch 4/50
Epoch 00003: val_loss did not improve
2s - loss: 0.0197 - acc: 0.5374 - mean_squared_error: 0.0197 - val_loss: 0.
0097 - val_acc: 0.6963 - val_mean_squared_error: 0.0097
Epoch 5/50
Epoch 00004: val_loss did not improve
2s - loss: 0.0187 - acc: 0.5456 - mean_squared_error: 0.0187 - val_loss: 0.
0089 - val_acc: 0.6963 - val_mean_squared_error: 0.0089
Epoch 6/50
Epoch 00005: val_loss did not improve
2s - loss: 0.0194 - acc: 0.5660 - mean_squared_error: 0.0194 - val_loss: 0.
0067 - val_acc: 0.6963 - val_mean_squared_error: 0.0067
Epoch 7/50
Epoch 00006: val_loss did not improve
2s - loss: 0.0189 - acc: 0.5777 - mean_squared_error: 0.0189 - val_loss: 0.
0058 - val_acc: 0.6963 - val_mean_squared_error: 0.0058
Epoch 8/50
Epoch 00007: val_loss did not improve
2s - loss: 0.0196 - acc: 0.5672 - mean_squared_error: 0.0196 - val_loss: 0.
0060 - val_acc: 0.6963 - val_mean_squared_error: 0.0060
Epoch 9/50
Epoch 00008: val_loss improved from 0.00495 to 0.00423, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0179 - acc: 0.5783 - mean_squared_error: 0.0179 - val_loss: 0.
0042 - val_acc: 0.6963 - val_mean_squared_error: 0.0042
Epoch 10/50
Epoch 00009: val_loss did not improve
2s - loss: 0.0181 - acc: 0.5859 - mean_squared_error: 0.0181 - val_loss: 0.
0051 - val_acc: 0.6963 - val_mean_squared_error: 0.0051
Epoch 11/50
Epoch 00010: val_loss did not improve
2s - loss: 0.0179 - acc: 0.5707 - mean_squared_error: 0.0179 - val_loss: 0.
0072 - val_acc: 0.6963 - val_mean_squared_error: 0.0072
Epoch 12/50
Epoch 00011: val_loss improved from 0.00423 to 0.00418, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0184 - acc: 0.5613 - mean_squared_error: 0.0184 - val_loss: 0.
0042 - val_acc: 0.6963 - val_mean_squared_error: 0.0042
Epoch 13/50
Epoch 00012: val_loss did not improve
2s - loss: 0.0177 - acc: 0.5748 - mean_squared_error: 0.0177 - val_loss: 0.
0045 - val_acc: 0.6963 - val_mean_squared_error: 0.0045
Epoch 14/50
Epoch 00013: val_loss did not improve
2s - loss: 0.0177 - acc: 0.6075 - mean_squared_error: 0.0177 - val_loss: 0.
0055 - val_acc: 0.6963 - val_mean_squared_error: 0.0055
Epoch 15/50
Epoch 00014: val_loss did not improve
2s - loss: 0.0174 - acc: 0.5648 - mean_squared_error: 0.0174 - val_loss: 0.
0068 - val_acc: 0.6963 - val_mean_squared_error: 0.0068
Epoch 16/50
Epoch 00015: val_loss improved from 0.00418 to 0.00345, saving model to ./m
```

```
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0168 - acc: 0.5666 - mean_squared_error: 0.0168 - val_loss: 0.
0034 - val_acc: 0.6963 - val_mean_squared_error: 0.0034
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0171 - acc: 0.5549 - mean_squared_error: 0.0171 - val_loss: 0.
0040 - val_acc: 0.6963 - val_mean_squared_error: 0.0040
Epoch 18/50
Epoch 00017: val_loss did not improve
2s - loss: 0.0163 - acc: 0.5736 - mean_squared_error: 0.0163 - val_loss: 0.
0051 - val_acc: 0.6986 - val_mean_squared_error: 0.0051
Epoch 19/50
Epoch 00018: val_loss did not improve
2s - loss: 0.0168 - acc: 0.5730 - mean_squared_error: 0.0168 - val_loss: 0.
0076 - val_acc: 0.6963 - val_mean_squared_error: 0.0076
Epoch 20/50
Epoch 00019: val_loss did not improve
2s - loss: 0.0162 - acc: 0.5789 - mean_squared_error: 0.0162 - val_loss: 0.
0049 - val_acc: 0.6986 - val_mean_squared_error: 0.0049
Epoch 21/50
Epoch 00020: val_loss did not improve
2s - loss: 0.0157 - acc: 0.6016 - mean_squared_error: 0.0157 - val_loss: 0.
0041 - val_acc: 0.7009 - val_mean_squared_error: 0.0041
Epoch 22/50
Epoch 00021: val_loss did not improve
2s - loss: 0.0160 - acc: 0.6057 - mean_squared_error: 0.0160 - val_loss: 0.
0040 - val_acc: 0.7009 - val_mean_squared_error: 0.0040
Epoch 23/50
Epoch 00022: val_loss did not improve
2s - loss: 0.0148 - acc: 0.5847 - mean_squared_error: 0.0148 - val_loss: 0.
0046 - val_acc: 0.7056 - val_mean_squared_error: 0.0046
Epoch 24/50
Epoch 00023: val_loss improved from 0.00345 to 0.00322, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0141 - acc: 0.5777 - mean_squared_error: 0.0141 - val_loss: 0.
0032 - val_acc: 0.7056 - val_mean_squared_error: 0.0032
Epoch 25/50
Epoch 00024: val_loss improved from 0.00322 to 0.00283, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0144 - acc: 0.5999 - mean_squared_error: 0.0144 - val_loss: 0.
0028 - val_acc: 0.7009 - val_mean_squared_error: 0.0028
Epoch 26/50
Epoch 00025: val_loss did not improve
2s - loss: 0.0138 - acc: 0.6069 - mean_squared_error: 0.0138 - val_loss: 0.
0035 - val_acc: 0.7173 - val_mean_squared_error: 0.0035
Epoch 27/50
Epoch 00026: val_loss did not improve
2s - loss: 0.0140 - acc: 0.6338 - mean_squared_error: 0.0140 - val_loss: 0.
0035 - val_acc: 0.7009 - val_mean_squared_error: 0.0035
Epoch 28/50
Epoch 00027: val_loss did not improve
2s - loss: 0.0136 - acc: 0.6121 - mean_squared_error: 0.0136 - val_loss: 0.
0036 - val_acc: 0.7103 - val_mean_squared_error: 0.0036
Epoch 29/50
Epoch 00028: val_loss did not improve
2s - loss: 0.0139 - acc: 0.6139 - mean_squared_error: 0.0139 - val_loss: 0.
0033 - val_acc: 0.7173 - val_mean_squared_error: 0.0033
```

```
Epoch 30/50
Epoch 00029: val_loss did not improve
2s - loss: 0.0134 - acc: 0.6145 - mean_squared_error: 0.0134 - val_loss: 0.
0036 - val_acc: 0.7173 - val_mean_squared_error: 0.0036
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0137 - acc: 0.6016 - mean_squared_error: 0.0137 - val_loss: 0.
0030 - val_acc: 0.7079 - val_mean_squared_error: 0.0030
Epoch 32/50
Epoch 00031: val_loss did not improve
2s - loss: 0.0132 - acc: 0.6121 - mean_squared_error: 0.0132 - val_loss: 0.
0034 - val_acc: 0.7056 - val_mean_squared_error: 0.0034
Epoch 33/50
Epoch 00032: val_loss did not improve
2s - loss: 0.0134 - acc: 0.6291 - mean_squared_error: 0.0134 - val_loss: 0.
0046 - val_acc: 0.7126 - val_mean_squared_error: 0.0046
Epoch 34/50
Epoch 00033: val_loss improved from 0.00283 to 0.00279, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0131 - acc: 0.6116 - mean_squared_error: 0.0131 - val_loss: 0.
0028 - val_acc: 0.7126 - val_mean_squared_error: 0.0028
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0126 - acc: 0.6262 - mean_squared_error: 0.0126 - val_loss: 0.
0033 - val_acc: 0.7103 - val_mean_squared_error: 0.0033
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0131 - acc: 0.6145 - mean_squared_error: 0.0131 - val_loss: 0.
0030 - val_acc: 0.7150 - val_mean_squared_error: 0.0030
Epoch 37/50
Epoch 00036: val_loss did not improve
2s - loss: 0.0132 - acc: 0.6139 - mean_squared_error: 0.0132 - val_loss: 0.
0048 - val_acc: 0.7056 - val_mean_squared_error: 0.0048
Epoch 38/50
Epoch 00037: val_loss did not improve
2s - loss: 0.0126 - acc: 0.6285 - mean_squared_error: 0.0126 - val_loss: 0.
0042 - val_acc: 0.7056 - val_mean_squared_error: 0.0042
Epoch 39/50
Epoch 00038: val_loss improved from 0.00279 to 0.00278, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0124 - acc: 0.6279 - mean_squared_error: 0.0124 - val_loss: 0.
0028 - val_acc: 0.7056 - val_mean_squared_error: 0.0028
Epoch 40/50
Epoch 00039: val_loss did not improve
2s - loss: 0.0122 - acc: 0.6425 - mean_squared_error: 0.0122 - val_loss: 0.
0088 - val_acc: 0.7290 - val_mean_squared_error: 0.0088
Epoch 41/50
Epoch 00040: val_loss did not improve
2s - loss: 0.0121 - acc: 0.6273 - mean_squared_error: 0.0121 - val_loss: 0.
0029 - val_acc: 0.6986 - val_mean_squared_error: 0.0029
Epoch 42/50
Epoch 00041: val_loss did not improve
2s - loss: 0.0127 - acc: 0.6086 - mean_squared_error: 0.0127 - val_loss: 0.
0047 - val_acc: 0.7033 - val_mean_squared_error: 0.0047
Epoch 43/50
Epoch 00042: val_loss improved from 0.00278 to 0.00250, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
```

```
2s - loss: 0.0120 - acc: 0.6326 - mean_squared_error: 0.0120 - val_loss: 0.
0025 - val_acc: 0.7056 - val_mean_squared_error: 0.0025
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0119 - acc: 0.6314 - mean_squared_error: 0.0119 - val_loss: 0.
0036 - val_acc: 0.7079 - val_mean_squared_error: 0.0036
Epoch 45/50
Epoch 00044: val_loss improved from 0.00250 to 0.00232, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
2s - loss: 0.0118 - acc: 0.6227 - mean_squared_error: 0.0118 - val_loss: 0.
0023 - val_acc: 0.7033 - val_mean_squared_error: 0.0023
Epoch 46/50
Epoch 00045: val_loss did not improve
2s - loss: 0.0118 - acc: 0.6250 - mean_squared_error: 0.0118 - val_loss: 0.
0048 - val_acc: 0.7150 - val_mean_squared_error: 0.0048
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0117 - acc: 0.6303 - mean_squared_error: 0.0117 - val_loss: 0.
0030 - val_acc: 0.7243 - val_mean_squared_error: 0.0030
Epoch 48/50
Epoch 00047: val_loss did not improve
2s - loss: 0.0118 - acc: 0.6238 - mean_squared_error: 0.0118 - val_loss: 0.
0043 - val_acc: 0.7243 - val_mean_squared_error: 0.0043
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0117 - acc: 0.6157 - mean_squared_error: 0.0117 - val_loss: 0.
0025 - val_acc: 0.7196 - val_mean_squared_error: 0.0025
Epoch 50/50
Epoch 00049: val_loss did not improve
2s - loss: 0.0114 - acc: 0.6250 - mean_squared_error: 0.0114 - val_loss: 0.
0024 - val_acc: 0.7220 - val_mean_squared_error: 0.0024



  Running model: bigger_base_model w/opt: Adadelta
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.00761, saving model to ./mode
l/Adadelta_model.weights.best.hdf5
4s - loss: 0.0277 - acc: 0.3989 - mean_squared_error: 0.0277 - val_loss: 0.
0076 - val_acc: 0.6963 - val_mean_squared_error: 0.0076
Epoch 2/50
Epoch 00001: val_loss improved from 0.00761 to 0.00502, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0140 - acc: 0.4889 - mean_squared_error: 0.0140 - val_loss: 0.
0050 - val_acc: 0.6963 - val_mean_squared_error: 0.0050
Epoch 3/50
Epoch 00002: val_loss did not improve
2s - loss: 0.0106 - acc: 0.5356 - mean_squared_error: 0.0106 - val_loss: 0.
0070 - val_acc: 0.6963 - val_mean_squared_error: 0.0070
Epoch 4/50
Epoch 00003: val_loss improved from 0.00502 to 0.00420, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0093 - acc: 0.5625 - mean_squared_error: 0.0093 - val_loss: 0.
0042 - val_acc: 0.6963 - val_mean_squared_error: 0.0042
Epoch 5/50
Epoch 00004: val_loss did not improve
```

```
2s - loss: 0.0080 - acc: 0.6028 - mean_squared_error: 0.0080 - val_loss: 0.
0043 - val_acc: 0.6963 - val_mean_squared_error: 0.0043
Epoch 6/50
Epoch 00005: val_loss improved from 0.00420 to 0.00389, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0077 - acc: 0.6040 - mean_squared_error: 0.0077 - val_loss: 0.
0039 - val_acc: 0.6963 - val_mean_squared_error: 0.0039
Epoch 7/50
Epoch 00006: val_loss did not improve
2s - loss: 0.0070 - acc: 0.6238 - mean_squared_error: 0.0070 - val_loss: 0.
0061 - val_acc: 0.6963 - val_mean_squared_error: 0.0061
Epoch 8/50
Epoch 00007: val_loss did not improve
2s - loss: 0.0066 - acc: 0.6238 - mean_squared_error: 0.0066 - val_loss: 0.
0040 - val_acc: 0.6963 - val_mean_squared_error: 0.0040
Epoch 9/50
Epoch 00008: val_loss improved from 0.00389 to 0.00364, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0061 - acc: 0.6367 - mean_squared_error: 0.0061 - val_loss: 0.
0036 - val_acc: 0.6963 - val_mean_squared_error: 0.0036
Epoch 10/50
Epoch 00009: val_loss did not improve
2s - loss: 0.0059 - acc: 0.6589 - mean_squared_error: 0.0059 - val_loss: 0.
0042 - val_acc: 0.6963 - val_mean_squared_error: 0.0042
Epoch 11/50
Epoch 00010: val_loss did not improve
2s - loss: 0.0056 - acc: 0.6688 - mean_squared_error: 0.0056 - val_loss: 0.
0040 - val_acc: 0.6963 - val_mean_squared_error: 0.0040
Epoch 12/50
Epoch 00011: val_loss did not improve
2s - loss: 0.0053 - acc: 0.6641 - mean_squared_error: 0.0053 - val_loss: 0.
0037 - val_acc: 0.6963 - val_mean_squared_error: 0.0037
Epoch 13/50
Epoch 00012: val_loss improved from 0.00364 to 0.00306, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0051 - acc: 0.6776 - mean_squared_error: 0.0051 - val_loss: 0.
0031 - val_acc: 0.7009 - val_mean_squared_error: 0.0031
Epoch 14/50
Epoch 00013: val_loss did not improve
2s - loss: 0.0051 - acc: 0.6776 - mean_squared_error: 0.0051 - val_loss: 0.
0032 - val_acc: 0.7056 - val_mean_squared_error: 0.0032
Epoch 15/50
Epoch 00014: val_loss improved from 0.00306 to 0.00305, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0049 - acc: 0.6869 - mean_squared_error: 0.0049 - val_loss: 0.
0031 - val_acc: 0.7033 - val_mean_squared_error: 0.0031
Epoch 16/50
Epoch 00015: val_loss improved from 0.00305 to 0.00298, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0045 - acc: 0.6834 - mean_squared_error: 0.0045 - val_loss: 0.
0030 - val_acc: 0.6986 - val_mean_squared_error: 0.0030
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0044 - acc: 0.6723 - mean_squared_error: 0.0044 - val_loss: 0.
0043 - val_acc: 0.7056 - val_mean_squared_error: 0.0043
Epoch 18/50
Epoch 00017: val_loss improved from 0.00298 to 0.00263, saving model to ./m
```

```
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0043 - acc: 0.6893 - mean_squared_error: 0.0043 - val_loss: 0.
0026 - val_acc: 0.7103 - val_mean_squared_error: 0.0026
Epoch 19/50
Epoch 00018: val_loss improved from 0.00263 to 0.00248, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0041 - acc: 0.6840 - mean_squared_error: 0.0041 - val_loss: 0.
0025 - val_acc: 0.7079 - val_mean_squared_error: 0.0025
Epoch 20/50
Epoch 00019: val_loss improved from 0.00248 to 0.00241, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0040 - acc: 0.6793 - mean_squared_error: 0.0040 - val_loss: 0.
0024 - val_acc: 0.7150 - val_mean_squared_error: 0.0024
Epoch 21/50
Epoch 00020: val_loss did not improve
2s - loss: 0.0039 - acc: 0.6793 - mean_squared_error: 0.0039 - val_loss: 0.
0024 - val_acc: 0.7126 - val_mean_squared_error: 0.0024
Epoch 22/50
Epoch 00021: val_loss improved from 0.00241 to 0.00226, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0038 - acc: 0.6735 - mean_squared_error: 0.0038 - val_loss: 0.
0023 - val_acc: 0.7196 - val_mean_squared_error: 0.0023
Epoch 23/50
Epoch 00022: val_loss did not improve
2s - loss: 0.0037 - acc: 0.6928 - mean_squared_error: 0.0037 - val_loss: 0.
0023 - val_acc: 0.7243 - val_mean_squared_error: 0.0023
Epoch 24/50
Epoch 00023: val_loss improved from 0.00226 to 0.00223, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0036 - acc: 0.6881 - mean_squared_error: 0.0036 - val_loss: 0.
0022 - val_acc: 0.7243 - val_mean_squared_error: 0.0022
Epoch 25/50
Epoch 00024: val_loss improved from 0.00223 to 0.00222, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0035 - acc: 0.7068 - mean_squared_error: 0.0035 - val_loss: 0.
0022 - val_acc: 0.7173 - val_mean_squared_error: 0.0022
Epoch 26/50
Epoch 00025: val_loss did not improve
2s - loss: 0.0033 - acc: 0.6951 - mean_squared_error: 0.0033 - val_loss: 0.
0024 - val_acc: 0.7220 - val_mean_squared_error: 0.0024
Epoch 27/50
Epoch 00026: val_loss improved from 0.00222 to 0.00208, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0034 - acc: 0.6881 - mean_squared_error: 0.0034 - val_loss: 0.
0021 - val_acc: 0.7173 - val_mean_squared_error: 0.0021
Epoch 28/50
Epoch 00027: val_loss did not improve
2s - loss: 0.0033 - acc: 0.7103 - mean_squared_error: 0.0033 - val_loss: 0.
0022 - val_acc: 0.7196 - val_mean_squared_error: 0.0022
Epoch 29/50
Epoch 00028: val_loss did not improve
2s - loss: 0.0032 - acc: 0.7079 - mean_squared_error: 0.0032 - val_loss: 0.
0021 - val_acc: 0.7383 - val_mean_squared_error: 0.0021
Epoch 30/50
Epoch 00029: val_loss improved from 0.00208 to 0.00206, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0032 - acc: 0.7144 - mean_squared_error: 0.0032 - val_loss: 0.
```

```
0021 - val_acc: 0.7360 - val_mean_squared_error: 0.0021
Epoch 31/50
Epoch 00030: val_loss improved from 0.00206 to 0.00202, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0030 - acc: 0.7021 - mean_squared_error: 0.0030 - val_loss: 0.
0020 - val_acc: 0.7196 - val_mean_squared_error: 0.0020
Epoch 32/50
Epoch 00031: val_loss did not improve
2s - loss: 0.0030 - acc: 0.7144 - mean_squared_error: 0.0030 - val_loss: 0.
0024 - val_acc: 0.7360 - val_mean_squared_error: 0.0024
Epoch 33/50
Epoch 00032: val_loss improved from 0.00202 to 0.00202, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0030 - acc: 0.7126 - mean_squared_error: 0.0030 - val_loss: 0.
0020 - val_acc: 0.7360 - val_mean_squared_error: 0.0020
Epoch 34/50
Epoch 00033: val_loss improved from 0.00202 to 0.00193, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0029 - acc: 0.7056 - mean_squared_error: 0.0029 - val_loss: 0.
0019 - val_acc: 0.7266 - val_mean_squared_error: 0.0019
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0029 - acc: 0.7091 - mean_squared_error: 0.0029 - val_loss: 0.
0020 - val_acc: 0.7336 - val_mean_squared_error: 0.0020
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0028 - acc: 0.7231 - mean_squared_error: 0.0028 - val_loss: 0.
0023 - val_acc: 0.7430 - val_mean_squared_error: 0.0023
Epoch 37/50
Epoch 00036: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7033 - mean_squared_error: 0.0027 - val_loss: 0.
0020 - val_acc: 0.7336 - val_mean_squared_error: 0.0020
Epoch 38/50
Epoch 00037: val_loss improved from 0.00193 to 0.00183, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0027 - acc: 0.7208 - mean_squared_error: 0.0027 - val_loss: 0.
0018 - val_acc: 0.7266 - val_mean_squared_error: 0.0018
Epoch 39/50
Epoch 00038: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7185 - mean_squared_error: 0.0027 - val_loss: 0.
0020 - val_acc: 0.7383 - val_mean_squared_error: 0.0020
Epoch 40/50
Epoch 00039: val_loss did not improve
2s - loss: 0.0026 - acc: 0.7196 - mean_squared_error: 0.0026 - val_loss: 0.
0018 - val_acc: 0.7407 - val_mean_squared_error: 0.0018
Epoch 41/50
Epoch 00040: val_loss improved from 0.00183 to 0.00173, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0026 - acc: 0.7173 - mean_squared_error: 0.0026 - val_loss: 0.
0017 - val_acc: 0.7383 - val_mean_squared_error: 0.0017
Epoch 42/50
Epoch 00041: val_loss did not improve
2s - loss: 0.0025 - acc: 0.7138 - mean_squared_error: 0.0025 - val_loss: 0.
0023 - val_acc: 0.7360 - val_mean_squared_error: 0.0023
Epoch 43/50
Epoch 00042: val_loss did not improve
2s - loss: 0.0025 - acc: 0.7261 - mean_squared_error: 0.0025 - val_loss: 0.
```

```
0017 - val_acc: 0.7336 - val_mean_squared_error: 0.0017
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0025 - acc: 0.7208 - mean_squared_error: 0.0025 - val_loss: 0.
0017 - val_acc: 0.7243 - val_mean_squared_error: 0.0017
Epoch 45/50
Epoch 00044: val_loss improved from 0.00173 to 0.00173, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0025 - acc: 0.7185 - mean_squared_error: 0.0025 - val_loss: 0.
0017 - val_acc: 0.7407 - val_mean_squared_error: 0.0017
Epoch 46/50
Epoch 00045: val_loss did not improve
2s - loss: 0.0024 - acc: 0.7249 - mean_squared_error: 0.0024 - val_loss: 0.
0018 - val_acc: 0.7360 - val_mean_squared_error: 0.0018
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0024 - acc: 0.7290 - mean_squared_error: 0.0024 - val_loss: 0.
0019 - val_acc: 0.7336 - val_mean_squared_error: 0.0019
Epoch 48/50
Epoch 00047: val_loss improved from 0.00173 to 0.00166, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0024 - acc: 0.7161 - mean_squared_error: 0.0024 - val_loss: 0.
0017 - val_acc: 0.7383 - val_mean_squared_error: 0.0017
Epoch 49/50
Epoch 00048: val_loss improved from 0.00166 to 0.00165, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
3s - loss: 0.0023 - acc: 0.7214 - mean_squared_error: 0.0023 - val_loss: 0.
0017 - val_acc: 0.7336 - val_mean_squared_error: 0.0017
Epoch 50/50
Epoch 00049: val_loss did not improve
2s - loss: 0.0023 - acc: 0.7196 - mean_squared_error: 0.0023 - val_loss: 0.
0019 - val_acc: 0.7453 - val_mean_squared_error: 0.0019



  Running model: bigger_base_model w/opt: Adam
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.00470, saving model to ./mode
l/Adam_model.weights.best.hdf5
4s - loss: 0.0233 - acc: 0.4544 - mean_squared_error: 0.0233 - val_loss: 0.
0047 - val_acc: 0.6963 - val_mean_squared_error: 0.0047
Epoch 2/50
Epoch 00001: val_loss improved from 0.00470 to 0.00394, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0083 - acc: 0.6005 - mean_squared_error: 0.0083 - val_loss: 0.
0039 - val_acc: 0.6963 - val_mean_squared_error: 0.0039
Epoch 3/50
Epoch 00002: val_loss improved from 0.00394 to 0.00285, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0064 - acc: 0.6279 - mean_squared_error: 0.0064 - val_loss: 0.
0029 - val_acc: 0.7009 - val_mean_squared_error: 0.0029
Epoch 4/50
Epoch 00003: val_loss improved from 0.00285 to 0.00249, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0053 - acc: 0.6548 - mean_squared_error: 0.0053 - val_loss: 0.
0025 - val_acc: 0.7079 - val_mean_squared_error: 0.0025
```

```
Epoch 5/50
Epoch 00004: val_loss improved from 0.00249 to 0.00216, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0044 - acc: 0.6641 - mean_squared_error: 0.0044 - val_loss: 0.
0022 - val_acc: 0.7079 - val_mean_squared_error: 0.0022
Epoch 6/50
Epoch 00005: val_loss did not improve
2s - loss: 0.0041 - acc: 0.6735 - mean_squared_error: 0.0041 - val_loss: 0.
0037 - val_acc: 0.7290 - val_mean_squared_error: 0.0037
Epoch 7/50
Epoch 00006: val_loss improved from 0.00216 to 0.00187, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0038 - acc: 0.6799 - mean_squared_error: 0.0038 - val_loss: 0.
0019 - val_acc: 0.7243 - val_mean_squared_error: 0.0019
Epoch 8/50
Epoch 00007: val_loss did not improve
2s - loss: 0.0034 - acc: 0.7039 - mean_squared_error: 0.0034 - val_loss: 0.
0019 - val_acc: 0.7383 - val_mean_squared_error: 0.0019
Epoch 9/50
Epoch 00008: val_loss did not improve
2s - loss: 0.0033 - acc: 0.7027 - mean_squared_error: 0.0033 - val_loss: 0.
0019 - val_acc: 0.7220 - val_mean_squared_error: 0.0019
Epoch 10/50
Epoch 00009: val_loss improved from 0.00187 to 0.00160, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0030 - acc: 0.7313 - mean_squared_error: 0.0030 - val_loss: 0.
0016 - val_acc: 0.7243 - val_mean_squared_error: 0.0016
Epoch 11/50
Epoch 00010: val_loss improved from 0.00160 to 0.00159, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0029 - acc: 0.7185 - mean_squared_error: 0.0029 - val_loss: 0.
0016 - val_acc: 0.7336 - val_mean_squared_error: 0.0016
Epoch 12/50
Epoch 00011: val_loss improved from 0.00159 to 0.00158, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0028 - acc: 0.7120 - mean_squared_error: 0.0028 - val_loss: 0.
0016 - val_acc: 0.7710 - val_mean_squared_error: 0.0016
Epoch 13/50
Epoch 00012: val_loss did not improve
2s - loss: 0.0028 - acc: 0.7331 - mean_squared_error: 0.0028 - val_loss: 0.
0016 - val_acc: 0.7710 - val_mean_squared_error: 0.0016
Epoch 14/50
Epoch 00013: val_loss did not improve
2s - loss: 0.0026 - acc: 0.7436 - mean_squared_error: 0.0026 - val_loss: 0.
0019 - val_acc: 0.7710 - val_mean_squared_error: 0.0019
Epoch 15/50
Epoch 00014: val_loss did not improve
2s - loss: 0.0025 - acc: 0.7401 - mean_squared_error: 0.0025 - val_loss: 0.
0018 - val_acc: 0.7523 - val_mean_squared_error: 0.0018
Epoch 16/50
Epoch 00015: val_loss improved from 0.00158 to 0.00141, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0025 - acc: 0.7319 - mean_squared_error: 0.0025 - val_loss: 0.
0014 - val_acc: 0.7640 - val_mean_squared_error: 0.0014
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0023 - acc: 0.7389 - mean_squared_error: 0.0023 - val_loss: 0.
```

```
0016 - val_acc: 0.7710 - val_mean_squared_error: 0.0016
Epoch 18/50
Epoch 00017: val_loss did not improve
2s - loss: 0.0022 - acc: 0.7389 - mean_squared_error: 0.0022 - val_loss: 0.
0015 - val_acc: 0.7523 - val_mean_squared_error: 0.0015
Epoch 19/50
Epoch 00018: val_loss did not improve
2s - loss: 0.0021 - acc: 0.7529 - mean_squared_error: 0.0021 - val_loss: 0.
0014 - val_acc: 0.7523 - val_mean_squared_error: 0.0014
Epoch 20/50
Epoch 00019: val_loss did not improve
2s - loss: 0.0021 - acc: 0.7523 - mean_squared_error: 0.0021 - val_loss: 0.
0015 - val_acc: 0.7710 - val_mean_squared_error: 0.0015
Epoch 21/50
Epoch 00020: val_loss did not improve
2s - loss: 0.0021 - acc: 0.7588 - mean_squared_error: 0.0021 - val_loss: 0.
0015 - val_acc: 0.7710 - val_mean_squared_error: 0.0015
Epoch 22/50
Epoch 00021: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7570 - mean_squared_error: 0.0019 - val_loss: 0.
0017 - val_acc: 0.7757 - val_mean_squared_error: 0.0017
Epoch 23/50
Epoch 00022: val_loss improved from 0.00141 to 0.00130, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0019 - acc: 0.7623 - mean_squared_error: 0.0019 - val_loss: 0.
0013 - val_acc: 0.7850 - val_mean_squared_error: 0.0013
Epoch 24/50
Epoch 00023: val_loss improved from 0.00130 to 0.00129, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0019 - acc: 0.7669 - mean_squared_error: 0.0019 - val_loss: 0.
0013 - val_acc: 0.7687 - val_mean_squared_error: 0.0013
Epoch 25/50
Epoch 00024: val_loss improved from 0.00129 to 0.00126, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0018 - acc: 0.7629 - mean_squared_error: 0.0018 - val_loss: 0.
0013 - val_acc: 0.7640 - val_mean_squared_error: 0.0013
Epoch 26/50
Epoch 00025: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7535 - mean_squared_error: 0.0019 - val_loss: 0.
0014 - val_acc: 0.7874 - val_mean_squared_error: 0.0014
Epoch 27/50
Epoch 00026: val_loss improved from 0.00126 to 0.00121, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0017 - acc: 0.7646 - mean_squared_error: 0.0017 - val_loss: 0.
0012 - val_acc: 0.7570 - val_mean_squared_error: 0.0012
Epoch 28/50
Epoch 00027: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7792 - mean_squared_error: 0.0016 - val_loss: 0.
0012 - val_acc: 0.7757 - val_mean_squared_error: 0.0012
Epoch 29/50
Epoch 00028: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7763 - mean_squared_error: 0.0016 - val_loss: 0.
0012 - val_acc: 0.7710 - val_mean_squared_error: 0.0012
Epoch 30/50
Epoch 00029: val_loss improved from 0.00121 to 0.00114, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0016 - acc: 0.7728 - mean_squared_error: 0.0016 - val_loss: 0.
```

```
0011 - val_acc: 0.8037 - val_mean_squared_error: 0.0011
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7769 - mean_squared_error: 0.0016 - val_loss: 0.
0012 - val_acc: 0.7967 - val_mean_squared_error: 0.0012
Epoch 32/50
Epoch 00031: val_loss improved from 0.00114 to 0.00114, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0016 - acc: 0.7769 - mean_squared_error: 0.0016 - val_loss: 0.
0011 - val_acc: 0.7944 - val_mean_squared_error: 0.0011
Epoch 33/50
Epoch 00032: val_loss improved from 0.00114 to 0.00111, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0015 - acc: 0.7751 - mean_squared_error: 0.0015 - val_loss: 0.
0011 - val_acc: 0.7967 - val_mean_squared_error: 0.0011
Epoch 34/50
Epoch 00033: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7862 - mean_squared_error: 0.0015 - val_loss: 0.
0011 - val_acc: 0.7991 - val_mean_squared_error: 0.0011
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0015 - acc: 0.7915 - mean_squared_error: 0.0015 - val_loss: 0.
0012 - val_acc: 0.7897 - val_mean_squared_error: 0.0012
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7856 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7921 - val_mean_squared_error: 0.0012
Epoch 37/50
Epoch 00036: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7751 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.8037 - val_mean_squared_error: 0.0012
Epoch 38/50
Epoch 00037: val_loss improved from 0.00111 to 0.00109, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0014 - acc: 0.7751 - mean_squared_error: 0.0014 - val_loss: 0.
0011 - val_acc: 0.8014 - val_mean_squared_error: 0.0011
Epoch 39/50
Epoch 00038: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7821 - mean_squared_error: 0.0013 - val_loss: 0.
0011 - val_acc: 0.8131 - val_mean_squared_error: 0.0011
Epoch 40/50
Epoch 00039: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7961 - mean_squared_error: 0.0013 - val_loss: 0.
0011 - val_acc: 0.8131 - val_mean_squared_error: 0.0011
Epoch 41/50
Epoch 00040: val_loss improved from 0.00109 to 0.00109, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0013 - acc: 0.7915 - mean_squared_error: 0.0013 - val_loss: 0.
0011 - val_acc: 0.8061 - val_mean_squared_error: 0.0011
Epoch 42/50
Epoch 00041: val_loss did not improve
2s - loss: 0.0012 - acc: 0.8049 - mean_squared_error: 0.0012 - val_loss: 0.
0011 - val_acc: 0.7897 - val_mean_squared_error: 0.0011
Epoch 43/50
Epoch 00042: val_loss improved from 0.00109 to 0.00106, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0013 - acc: 0.7921 - mean_squared_error: 0.0013 - val_loss: 0.
```

```
0011 - val_acc: 0.7991 - val_mean_squared_error: 0.0011
Epoch 44/50
Epoch 00043: val_loss improved from 0.00106 to 0.00104, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0012 - acc: 0.8119 - mean_squared_error: 0.0012 - val_loss: 0.
0010 - val_acc: 0.8248 - val_mean_squared_error: 0.0010
Epoch 45/50
Epoch 00044: val_loss did not improve
2s - loss: 0.0012 - acc: 0.8067 - mean_squared_error: 0.0012 - val_loss: 0.
0011 - val_acc: 0.7874 - val_mean_squared_error: 0.0011
Epoch 46/50
Epoch 00045: val_loss improved from 0.00104 to 0.00101, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0012 - acc: 0.7996 - mean_squared_error: 0.0012 - val_loss: 0.
0010 - val_acc: 0.8107 - val_mean_squared_error: 0.0010
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0011 - acc: 0.7880 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.7991 - val_mean_squared_error: 0.0011
Epoch 48/50
Epoch 00047: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8037 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.8014 - val_mean_squared_error: 0.0011
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8008 - mean_squared_error: 0.0011 - val_loss: 0.
0011 - val_acc: 0.8084 - val_mean_squared_error: 0.0011
Epoch 50/50
Epoch 00049: val_loss improved from 0.00101 to 0.00101, saving model to ./m
odel/Adam_model.weights.best.hdf5
3s - loss: 0.0011 - acc: 0.8026 - mean_squared_error: 0.0011 - val_loss: 0.
0010 - val_acc: 0.8224 - val_mean_squared_error: 0.0010



  Running model: bigger_base_model w/opt: Adamax
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.00483, saving model to ./mode
l/Adamax_model.weights.best.hdf5
4s - loss: 0.0345 - acc: 0.4118 - mean_squared_error: 0.0345 - val_loss: 0.
0048 - val_acc: 0.6893 - val_mean_squared_error: 0.0048
Epoch 2/50
Epoch 00001: val_loss improved from 0.00483 to 0.00428, saving model to ./m
odel/Adamax_model.weights.best.hdf5
3s - loss: 0.0117 - acc: 0.4790 - mean_squared_error: 0.0117 - val_loss: 0.
0043 - val_acc: 0.7033 - val_mean_squared_error: 0.0043
Epoch 3/50
Epoch 00002: val_loss improved from 0.00428 to 0.00411, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0096 - acc: 0.5409 - mean_squared_error: 0.0096 - val_loss: 0.
0041 - val_acc: 0.7103 - val_mean_squared_error: 0.0041
Epoch 4/50
Epoch 00003: val_loss did not improve
2s - loss: 0.0084 - acc: 0.5718 - mean_squared_error: 0.0084 - val_loss: 0.
0045 - val_acc: 0.7056 - val_mean_squared_error: 0.0045
Epoch 5/50
```

```
Epoch 00004: val_loss improved from 0.00411 to 0.00276, saving model to ./m
odel/Adamax_model.weights.best.hdf5
3s - loss: 0.0072 - acc: 0.5905 - mean_squared_error: 0.0072 - val_loss: 0.
0028 - val_acc: 0.7126 - val_mean_squared_error: 0.0028
Epoch 6/50
Epoch 00005: val_loss improved from 0.00276 to 0.00254, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0064 - acc: 0.6051 - mean_squared_error: 0.0064 - val_loss: 0.
0025 - val_acc: 0.7196 - val_mean_squared_error: 0.0025
Epoch 7/50
Epoch 00006: val_loss improved from 0.00254 to 0.00217, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0058 - acc: 0.6250 - mean_squared_error: 0.0058 - val_loss: 0.
0022 - val_acc: 0.7360 - val_mean_squared_error: 0.0022
Epoch 8/50
Epoch 00007: val_loss did not improve
2s - loss: 0.0055 - acc: 0.6314 - mean_squared_error: 0.0055 - val_loss: 0.
0029 - val_acc: 0.7126 - val_mean_squared_error: 0.0029
Epoch 9/50
Epoch 00008: val_loss did not improve
2s - loss: 0.0052 - acc: 0.6449 - mean_squared_error: 0.0052 - val_loss: 0.
0023 - val_acc: 0.7243 - val_mean_squared_error: 0.0023
Epoch 10/50
Epoch 00009: val_loss did not improve
2s - loss: 0.0050 - acc: 0.6507 - mean_squared_error: 0.0050 - val_loss: 0.
0026 - val_acc: 0.7477 - val_mean_squared_error: 0.0026
Epoch 11/50
Epoch 00010: val_loss improved from 0.00217 to 0.00212, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0049 - acc: 0.6606 - mean_squared_error: 0.0049 - val_loss: 0.
0021 - val_acc: 0.7430 - val_mean_squared_error: 0.0021
Epoch 12/50
Epoch 00011: val_loss improved from 0.00212 to 0.00186, saving model to ./m
odel/Adamax_model.weights.best.hdf5
3s - loss: 0.0047 - acc: 0.6700 - mean_squared_error: 0.0047 - val_loss: 0.
0019 - val_acc: 0.7336 - val_mean_squared_error: 0.0019
Epoch 13/50
Epoch 00012: val_loss did not improve
2s - loss: 0.0044 - acc: 0.6723 - mean_squared_error: 0.0044 - val_loss: 0.
0019 - val_acc: 0.7523 - val_mean_squared_error: 0.0019
Epoch 14/50
Epoch 00013: val_loss did not improve
2s - loss: 0.0041 - acc: 0.6840 - mean_squared_error: 0.0041 - val_loss: 0.
0019 - val_acc: 0.7453 - val_mean_squared_error: 0.0019
Epoch 15/50
Epoch 00014: val_loss improved from 0.00186 to 0.00169, saving model to ./m
odel/Adamax_model.weights.best.hdf5
3s - loss: 0.0041 - acc: 0.6770 - mean_squared_error: 0.0041 - val_loss: 0.
0017 - val_acc: 0.7570 - val_mean_squared_error: 0.0017
Epoch 16/50
Epoch 00015: val_loss did not improve
2s - loss: 0.0039 - acc: 0.6805 - mean_squared_error: 0.0039 - val_loss: 0.
0018 - val_acc: 0.7336 - val_mean_squared_error: 0.0018
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0039 - acc: 0.7085 - mean_squared_error: 0.0039 - val_loss: 0.
0018 - val_acc: 0.7570 - val_mean_squared_error: 0.0018
```

```
Epoch 18/50
Epoch 00017: val_loss improved from 0.00169 to 0.00151, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0040 - acc: 0.7039 - mean_squared_error: 0.0040 - val_loss: 0.
0015 - val_acc: 0.7710 - val_mean_squared_error: 0.0015
Epoch 19/50
Epoch 00018: val_loss did not improve
2s - loss: 0.0037 - acc: 0.7009 - mean_squared_error: 0.0037 - val_loss: 0.
0017 - val_acc: 0.7453 - val_mean_squared_error: 0.0017
Epoch 20/50
Epoch 00019: val_loss did not improve
2s - loss: 0.0036 - acc: 0.7062 - mean_squared_error: 0.0036 - val_loss: 0.
0019 - val_acc: 0.7453 - val_mean_squared_error: 0.0019
Epoch 21/50
Epoch 00020: val_loss did not improve
2s - loss: 0.0034 - acc: 0.7068 - mean_squared_error: 0.0034 - val_loss: 0.
0016 - val_acc: 0.7617 - val_mean_squared_error: 0.0016
Epoch 22/50
Epoch 00021: val_loss improved from 0.00151 to 0.00147, saving model to ./m
odel/Adamax_model.weights.best.hdf5
3s - loss: 0.0034 - acc: 0.7044 - mean_squared_error: 0.0034 - val_loss: 0.
0015 - val_acc: 0.7664 - val_mean_squared_error: 0.0015
Epoch 23/50
Epoch 00022: val_loss improved from 0.00147 to 0.00144, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0032 - acc: 0.7120 - mean_squared_error: 0.0032 - val_loss: 0.
0014 - val_acc: 0.7710 - val_mean_squared_error: 0.0014
Epoch 24/50
Epoch 00023: val_loss did not improve
2s - loss: 0.0031 - acc: 0.7091 - mean_squared_error: 0.0031 - val_loss: 0.
0016 - val_acc: 0.7664 - val_mean_squared_error: 0.0016
Epoch 25/50
Epoch 00024: val_loss did not improve
2s - loss: 0.0030 - acc: 0.7161 - mean_squared_error: 0.0030 - val_loss: 0.
0017 - val_acc: 0.7640 - val_mean_squared_error: 0.0017
Epoch 26/50
Epoch 00025: val_loss improved from 0.00144 to 0.00138, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0030 - acc: 0.7383 - mean_squared_error: 0.0030 - val_loss: 0.
0014 - val_acc: 0.7780 - val_mean_squared_error: 0.0014
Epoch 27/50
Epoch 00026: val_loss did not improve
2s - loss: 0.0028 - acc: 0.7360 - mean_squared_error: 0.0028 - val_loss: 0.
0016 - val_acc: 0.7780 - val_mean_squared_error: 0.0016
Epoch 28/50
Epoch 00027: val_loss did not improve
2s - loss: 0.0028 - acc: 0.7196 - mean_squared_error: 0.0028 - val_loss: 0.
0014 - val_acc: 0.7757 - val_mean_squared_error: 0.0014
Epoch 29/50
Epoch 00028: val_loss did not improve
2s - loss: 0.0028 - acc: 0.7348 - mean_squared_error: 0.0028 - val_loss: 0.
0014 - val_acc: 0.7640 - val_mean_squared_error: 0.0014
Epoch 30/50
Epoch 00029: val_loss improved from 0.00138 to 0.00131, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0027 - acc: 0.7301 - mean_squared_error: 0.0027 - val_loss: 0.
0013 - val_acc: 0.7664 - val_mean_squared_error: 0.0013
```

```
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0025 - acc: 0.7558 - mean_squared_error: 0.0025 - val_loss: 0.
0015 - val_acc: 0.7664 - val_mean_squared_error: 0.0015
Epoch 32/50
Epoch 00031: val_loss did not improve
2s - loss: 0.0026 - acc: 0.7430 - mean_squared_error: 0.0026 - val_loss: 0.
0013 - val_acc: 0.7640 - val_mean_squared_error: 0.0013
Epoch 33/50
Epoch 00032: val_loss did not improve
2s - loss: 0.0024 - acc: 0.7383 - mean_squared_error: 0.0024 - val_loss: 0.
0014 - val_acc: 0.7570 - val_mean_squared_error: 0.0014
Epoch 34/50
Epoch 00033: val_loss did not improve
2s - loss: 0.0024 - acc: 0.7395 - mean_squared_error: 0.0024 - val_loss: 0.
0014 - val_acc: 0.7570 - val_mean_squared_error: 0.0014
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0023 - acc: 0.7593 - mean_squared_error: 0.0023 - val_loss: 0.
0013 - val_acc: 0.7827 - val_mean_squared_error: 0.0013
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0024 - acc: 0.7389 - mean_squared_error: 0.0024 - val_loss: 0.
0014 - val_acc: 0.7547 - val_mean_squared_error: 0.0014
Epoch 37/50
Epoch 00036: val_loss did not improve
2s - loss: 0.0022 - acc: 0.7500 - mean_squared_error: 0.0022 - val_loss: 0.
0014 - val_acc: 0.7780 - val_mean_squared_error: 0.0014
Epoch 38/50
Epoch 00037: val_loss did not improve
2s - loss: 0.0022 - acc: 0.7500 - mean_squared_error: 0.0022 - val_loss: 0.
0014 - val_acc: 0.7827 - val_mean_squared_error: 0.0014
Epoch 39/50
Epoch 00038: val_loss improved from 0.00131 to 0.00127, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0021 - acc: 0.7518 - mean_squared_error: 0.0021 - val_loss: 0.
0013 - val_acc: 0.7874 - val_mean_squared_error: 0.0013
Epoch 40/50
Epoch 00039: val_loss improved from 0.00127 to 0.00119, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0021 - acc: 0.7576 - mean_squared_error: 0.0021 - val_loss: 0.
0012 - val_acc: 0.7687 - val_mean_squared_error: 0.0012
Epoch 41/50
Epoch 00040: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7523 - mean_squared_error: 0.0019 - val_loss: 0.
0012 - val_acc: 0.7687 - val_mean_squared_error: 0.0012
Epoch 42/50
Epoch 00041: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7769 - mean_squared_error: 0.0019 - val_loss: 0.
0014 - val_acc: 0.7850 - val_mean_squared_error: 0.0014
Epoch 43/50
Epoch 00042: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7518 - mean_squared_error: 0.0019 - val_loss: 0.
0013 - val_acc: 0.7687 - val_mean_squared_error: 0.0013
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7710 - mean_squared_error: 0.0018 - val_loss: 0.
```

```
0014 - val_acc: 0.7780 - val_mean_squared_error: 0.0014
Epoch 45/50
Epoch 00044: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7693 - mean_squared_error: 0.0018 - val_loss: 0.
0013 - val_acc: 0.7804 - val_mean_squared_error: 0.0013
Epoch 46/50
Epoch 00045: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7681 - mean_squared_error: 0.0017 - val_loss: 0.
0012 - val_acc: 0.7944 - val_mean_squared_error: 0.0012
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7716 - mean_squared_error: 0.0017 - val_loss: 0.
0012 - val_acc: 0.7897 - val_mean_squared_error: 0.0012
Epoch 48/50
Epoch 00047: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7915 - mean_squared_error: 0.0016 - val_loss: 0.
0014 - val_acc: 0.8037 - val_mean_squared_error: 0.0014
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7745 - mean_squared_error: 0.0016 - val_loss: 0.
0017 - val_acc: 0.7804 - val_mean_squared_error: 0.0017
Epoch 50/50
Epoch 00049: val_loss improved from 0.00119 to 0.00119, saving model to ./m
odel/Adamax_model.weights.best.hdf5
2s - loss: 0.0016 - acc: 0.7880 - mean_squared_error: 0.0016 - val_loss: 0.
0012 - val_acc: 0.7897 - val_mean_squared_error: 0.0012



  Running model: bigger_base_model w/opt: Nadam
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.00923, saving model to ./mode
l/Nadam_model.weights.best.hdf5
4s - loss: 0.1205 - acc: 0.4013 - mean_squared_error: 0.1205 - val_loss: 0.
0092 - val_acc: 0.6963 - val_mean_squared_error: 0.0092
Epoch 2/50
Epoch 00001: val_loss improved from 0.00923 to 0.00519, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0138 - acc: 0.5169 - mean_squared_error: 0.0138 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 3/50
Epoch 00002: val_loss did not improve
2s - loss: 0.0106 - acc: 0.5543 - mean_squared_error: 0.0106 - val_loss: 0.
0094 - val_acc: 0.6963 - val_mean_squared_error: 0.0094
Epoch 4/50
Epoch 00003: val_loss did not improve
2s - loss: 0.0095 - acc: 0.5970 - mean_squared_error: 0.0095 - val_loss: 0.
0074 - val_acc: 0.6963 - val_mean_squared_error: 0.0074
Epoch 5/50
Epoch 00004: val_loss improved from 0.00519 to 0.00326, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0081 - acc: 0.6016 - mean_squared_error: 0.0081 - val_loss: 0.
0033 - val_acc: 0.6963 - val_mean_squared_error: 0.0033
Epoch 6/50
Epoch 00005: val_loss improved from 0.00326 to 0.00294, saving model to ./m
odel/Nadam_model.weights.best.hdf5
```

```
3s - loss: 0.0069 - acc: 0.6197 - mean_squared_error: 0.0069 - val_loss: 0.
0029 - val_acc: 0.6986 - val_mean_squared_error: 0.0029
Epoch 7/50
Epoch 00006: val_loss improved from 0.00294 to 0.00250, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0067 - acc: 0.6338 - mean_squared_error: 0.0067 - val_loss: 0.
0025 - val_acc: 0.7196 - val_mean_squared_error: 0.0025
Epoch 8/50
Epoch 00007: val_loss did not improve
2s - loss: 0.0059 - acc: 0.6472 - mean_squared_error: 0.0059 - val_loss: 0.
0025 - val_acc: 0.7313 - val_mean_squared_error: 0.0025
Epoch 9/50
Epoch 00008: val_loss did not improve
2s - loss: 0.0056 - acc: 0.6612 - mean_squared_error: 0.0056 - val_loss: 0.
0073 - val_acc: 0.7243 - val_mean_squared_error: 0.0073
Epoch 10/50
Epoch 00009: val_loss did not improve
2s - loss: 0.0050 - acc: 0.6828 - mean_squared_error: 0.0050 - val_loss: 0.
0026 - val_acc: 0.7570 - val_mean_squared_error: 0.0026
Epoch 11/50
Epoch 00010: val_loss improved from 0.00250 to 0.00183, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0047 - acc: 0.6782 - mean_squared_error: 0.0047 - val_loss: 0.
0018 - val_acc: 0.7710 - val_mean_squared_error: 0.0018
Epoch 12/50
Epoch 00011: val_loss did not improve
2s - loss: 0.0044 - acc: 0.6846 - mean_squared_error: 0.0044 - val_loss: 0.
0022 - val_acc: 0.7500 - val_mean_squared_error: 0.0022
Epoch 13/50
Epoch 00012: val_loss did not improve
2s - loss: 0.0041 - acc: 0.6857 - mean_squared_error: 0.0041 - val_loss: 0.
0021 - val_acc: 0.7430 - val_mean_squared_error: 0.0021
Epoch 14/50
Epoch 00013: val_loss did not improve
2s - loss: 0.0040 - acc: 0.6834 - mean_squared_error: 0.0040 - val_loss: 0.
0021 - val_acc: 0.7593 - val_mean_squared_error: 0.0021
Epoch 15/50
Epoch 00014: val_loss improved from 0.00183 to 0.00167, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0037 - acc: 0.6869 - mean_squared_error: 0.0037 - val_loss: 0.
0017 - val_acc: 0.7593 - val_mean_squared_error: 0.0017
Epoch 16/50
Epoch 00015: val_loss did not improve
2s - loss: 0.0034 - acc: 0.6998 - mean_squared_error: 0.0034 - val_loss: 0.
0017 - val_acc: 0.7570 - val_mean_squared_error: 0.0017
Epoch 17/50
Epoch 00016: val_loss did not improve
2s - loss: 0.0032 - acc: 0.7050 - mean_squared_error: 0.0032 - val_loss: 0.
0018 - val_acc: 0.7500 - val_mean_squared_error: 0.0018
Epoch 18/50
Epoch 00017: val_loss did not improve
2s - loss: 0.0031 - acc: 0.7278 - mean_squared_error: 0.0031 - val_loss: 0.
0026 - val_acc: 0.7150 - val_mean_squared_error: 0.0026
Epoch 19/50
Epoch 00018: val_loss did not improve
2s - loss: 0.0029 - acc: 0.7114 - mean_squared_error: 0.0029 - val_loss: 0.
0022 - val_acc: 0.7500 - val_mean_squared_error: 0.0022
```

```
Epoch 20/50
Epoch 00019: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7284 - mean_squared_error: 0.0027 - val_loss: 0.
0017 - val_acc: 0.7523 - val_mean_squared_error: 0.0017
Epoch 21/50
Epoch 00020: val_loss did not improve
2s - loss: 0.0027 - acc: 0.7196 - mean_squared_error: 0.0027 - val_loss: 0.
0020 - val_acc: 0.7850 - val_mean_squared_error: 0.0020
Epoch 22/50
Epoch 00021: val_loss did not improve
2s - loss: 0.0025 - acc: 0.7278 - mean_squared_error: 0.0025 - val_loss: 0.
0017 - val_acc: 0.7734 - val_mean_squared_error: 0.0017
Epoch 23/50
Epoch 00022: val_loss improved from 0.00167 to 0.00137, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0024 - acc: 0.7407 - mean_squared_error: 0.0024 - val_loss: 0.
0014 - val_acc: 0.7757 - val_mean_squared_error: 0.0014
Epoch 24/50
Epoch 00023: val_loss did not improve
2s - loss: 0.0023 - acc: 0.7383 - mean_squared_error: 0.0023 - val_loss: 0.
0015 - val_acc: 0.7664 - val_mean_squared_error: 0.0015
Epoch 25/50
Epoch 00024: val_loss did not improve
2s - loss: 0.0022 - acc: 0.7436 - mean_squared_error: 0.0022 - val_loss: 0.
0027 - val_acc: 0.7336 - val_mean_squared_error: 0.0027
Epoch 26/50
Epoch 00025: val_loss improved from 0.00137 to 0.00135, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0021 - acc: 0.7588 - mean_squared_error: 0.0021 - val_loss: 0.
0014 - val_acc: 0.7827 - val_mean_squared_error: 0.0014
Epoch 27/50
Epoch 00026: val_loss did not improve
2s - loss: 0.0020 - acc: 0.7512 - mean_squared_error: 0.0020 - val_loss: 0.
0016 - val_acc: 0.7664 - val_mean_squared_error: 0.0016
Epoch 28/50
Epoch 00027: val_loss improved from 0.00135 to 0.00127, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0019 - acc: 0.7675 - mean_squared_error: 0.0019 - val_loss: 0.
0013 - val_acc: 0.8037 - val_mean_squared_error: 0.0013
Epoch 29/50
Epoch 00028: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7716 - mean_squared_error: 0.0019 - val_loss: 0.
0013 - val_acc: 0.7944 - val_mean_squared_error: 0.0013
Epoch 30/50
Epoch 00029: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7640 - mean_squared_error: 0.0018 - val_loss: 0.
0015 - val_acc: 0.7804 - val_mean_squared_error: 0.0015
Epoch 31/50
Epoch 00030: val_loss did not improve
2s - loss: 0.0018 - acc: 0.7710 - mean_squared_error: 0.0018 - val_loss: 0.
0014 - val_acc: 0.8014 - val_mean_squared_error: 0.0014
Epoch 32/50
Epoch 00031: val_loss did not improve
2s - loss: 0.0017 - acc: 0.7576 - mean_squared_error: 0.0017 - val_loss: 0.
0018 - val_acc: 0.7617 - val_mean_squared_error: 0.0018
Epoch 33/50
Epoch 00032: val_loss did not improve
```

```
2s - loss: 0.0016 - acc: 0.7798 - mean_squared_error: 0.0016 - val_loss: 0.
0013 - val_acc: 0.7804 - val_mean_squared_error: 0.0013
Epoch 34/50
Epoch 00033: val_loss improved from 0.00127 to 0.00124, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0016 - acc: 0.7704 - mean_squared_error: 0.0016 - val_loss: 0.
0012 - val_acc: 0.7874 - val_mean_squared_error: 0.0012
Epoch 35/50
Epoch 00034: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7646 - mean_squared_error: 0.0016 - val_loss: 0.
0013 - val_acc: 0.7500 - val_mean_squared_error: 0.0013
Epoch 36/50
Epoch 00035: val_loss did not improve
2s - loss: 0.0019 - acc: 0.7658 - mean_squared_error: 0.0019 - val_loss: 0.
0014 - val_acc: 0.7687 - val_mean_squared_error: 0.0014
Epoch 37/50
Epoch 00036: val_loss did not improve
2s - loss: 0.0016 - acc: 0.7652 - mean_squared_error: 0.0016 - val_loss: 0.
0013 - val_acc: 0.7944 - val_mean_squared_error: 0.0013
Epoch 38/50
Epoch 00037: val_loss improved from 0.00124 to 0.00120, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0014 - acc: 0.7722 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7921 - val_mean_squared_error: 0.0012
Epoch 39/50
Epoch 00038: val_loss did not improve
2s - loss: 0.0014 - acc: 0.7745 - mean_squared_error: 0.0014 - val_loss: 0.
0013 - val_acc: 0.7897 - val_mean_squared_error: 0.0013
Epoch 40/50
Epoch 00039: val_loss improved from 0.00120 to 0.00116, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0014 - acc: 0.7850 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7921 - val_mean_squared_error: 0.0012
Epoch 41/50
Epoch 00040: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7903 - mean_squared_error: 0.0013 - val_loss: 0.
0012 - val_acc: 0.7921 - val_mean_squared_error: 0.0012
Epoch 42/50
Epoch 00041: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7932 - mean_squared_error: 0.0013 - val_loss: 0.
0013 - val_acc: 0.7734 - val_mean_squared_error: 0.0013
Epoch 43/50
Epoch 00042: val_loss did not improve
2s - loss: 0.0013 - acc: 0.7775 - mean_squared_error: 0.0013 - val_loss: 0.
0012 - val_acc: 0.7780 - val_mean_squared_error: 0.0012
Epoch 44/50
Epoch 00043: val_loss did not improve
2s - loss: 0.0012 - acc: 0.7839 - mean_squared_error: 0.0012 - val_loss: 0.
0016 - val_acc: 0.7687 - val_mean_squared_error: 0.0016
Epoch 45/50
Epoch 00044: val_loss did not improve
2s - loss: 0.0012 - acc: 0.7956 - mean_squared_error: 0.0012 - val_loss: 0.
0013 - val_acc: 0.7874 - val_mean_squared_error: 0.0013
Epoch 46/50
Epoch 00045: val_loss improved from 0.00116 to 0.00115, saving model to ./m
odel/Nadam_model.weights.best.hdf5
3s - loss: 0.0012 - acc: 0.7821 - mean_squared_error: 0.0012 - val_loss: 0.
```

```
0011 - val_acc: 0.7921 - val_mean_squared_error: 0.0011
Epoch 47/50
Epoch 00046: val_loss did not improve
2s - loss: 0.0012 - acc: 0.7897 - mean_squared_error: 0.0012 - val_loss: 0.
0012 - val_acc: 0.7991 - val_mean_squared_error: 0.0012
Epoch 48/50
Epoch 00047: val_loss did not improve
2s - loss: 0.0012 - acc: 0.8026 - mean_squared_error: 0.0012 - val_loss: 0.
0013 - val_acc: 0.7897 - val_mean_squared_error: 0.0013
Epoch 49/50
Epoch 00048: val_loss did not improve
2s - loss: 0.0012 - acc: 0.7979 - mean_squared_error: 0.0012 - val_loss: 0.
0012 - val_acc: 0.7921 - val_mean_squared_error: 0.0012
Epoch 50/50
Epoch 00049: val_loss did not improve
2s - loss: 0.0011 - acc: 0.8119 - mean_squared_error: 0.0011 - val_loss: 0.
0012 - val_acc: 0.7897 - val_mean_squared_error: 0.0012



  Running model: dropout_base_model w/opt: SGD
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.04077, saving model to ./mode
l/SGD_model.weights.best.hdf5
3s - loss: 0.0746 - acc: 0.3037 - mean_squared_error: 0.0746 - val_loss: 0.
0408 - val_acc: 0.6776 - val_mean_squared_error: 0.0408
Epoch 2/50
Epoch 00001: val_loss improved from 0.04077 to 0.03774, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0392 - acc: 0.3370 - mean_squared_error: 0.0392 - val_loss: 0.
0377 - val_acc: 0.6963 - val_mean_squared_error: 0.0377
Epoch 3/50
Epoch 00002: val_loss improved from 0.03774 to 0.03530, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0332 - acc: 0.3662 - mean_squared_error: 0.0332 - val_loss: 0.
0353 - val_acc: 0.6963 - val_mean_squared_error: 0.0353
Epoch 4/50
Epoch 00003: val_loss improved from 0.03530 to 0.03400, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0298 - acc: 0.3843 - mean_squared_error: 0.0298 - val_loss: 0.
0340 - val_acc: 0.6963 - val_mean_squared_error: 0.0340
Epoch 5/50
Epoch 00004: val_loss improved from 0.03400 to 0.03300, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0274 - acc: 0.3949 - mean_squared_error: 0.0274 - val_loss: 0.
0330 - val_acc: 0.6916 - val_mean_squared_error: 0.0330
Epoch 6/50
Epoch 00005: val_loss improved from 0.03300 to 0.03106, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0255 - acc: 0.4036 - mean_squared_error: 0.0255 - val_loss: 0.
0311 - val_acc: 0.6986 - val_mean_squared_error: 0.0311
Epoch 7/50
Epoch 00006: val_loss improved from 0.03106 to 0.03021, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0236 - acc: 0.3972 - mean_squared_error: 0.0236 - val_loss: 0.
0302 - val_acc: 0.6986 - val_mean_squared_error: 0.0302
```

```
Epoch 8/50
Epoch 00007: val_loss improved from 0.03021 to 0.02770, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0224 - acc: 0.3960 - mean_squared_error: 0.0224 - val_loss: 0.
0277 - val_acc: 0.6986 - val_mean_squared_error: 0.0277
Epoch 9/50
Epoch 00008: val_loss improved from 0.02770 to 0.02688, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0213 - acc: 0.4153 - mean_squared_error: 0.0213 - val_loss: 0.
0269 - val_acc: 0.6963 - val_mean_squared_error: 0.0269
Epoch 10/50
Epoch 00009: val_loss improved from 0.02688 to 0.02584, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0202 - acc: 0.4398 - mean_squared_error: 0.0202 - val_loss: 0.
0258 - val_acc: 0.6986 - val_mean_squared_error: 0.0258
Epoch 11/50
Epoch 00010: val_loss improved from 0.02584 to 0.02379, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0194 - acc: 0.4439 - mean_squared_error: 0.0194 - val_loss: 0.
0238 - val_acc: 0.6986 - val_mean_squared_error: 0.0238
Epoch 12/50
Epoch 00011: val_loss improved from 0.02379 to 0.02259, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0183 - acc: 0.4346 - mean_squared_error: 0.0183 - val_loss: 0.
0226 - val_acc: 0.6986 - val_mean_squared_error: 0.0226
Epoch 13/50
Epoch 00012: val_loss improved from 0.02259 to 0.02191, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0178 - acc: 0.4451 - mean_squared_error: 0.0178 - val_loss: 0.
0219 - val_acc: 0.6986 - val_mean_squared_error: 0.0219
Epoch 14/50
Epoch 00013: val_loss improved from 0.02191 to 0.02069, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0171 - acc: 0.4492 - mean_squared_error: 0.0171 - val_loss: 0.
0207 - val_acc: 0.6986 - val_mean_squared_error: 0.0207
Epoch 15/50
Epoch 00014: val_loss did not improve
1s - loss: 0.0166 - acc: 0.4363 - mean_squared_error: 0.0166 - val_loss: 0.
0207 - val_acc: 0.6986 - val_mean_squared_error: 0.0207
Epoch 16/50
Epoch 00015: val_loss improved from 0.02069 to 0.01867, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0158 - acc: 0.4486 - mean_squared_error: 0.0158 - val_loss: 0.
0187 - val_acc: 0.6986 - val_mean_squared_error: 0.0187
Epoch 17/50
Epoch 00016: val_loss improved from 0.01867 to 0.01818, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0154 - acc: 0.4544 - mean_squared_error: 0.0154 - val_loss: 0.
0182 - val_acc: 0.6986 - val_mean_squared_error: 0.0182
Epoch 18/50
Epoch 00017: val_loss improved from 0.01818 to 0.01765, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0153 - acc: 0.4457 - mean_squared_error: 0.0153 - val_loss: 0.
0177 - val_acc: 0.6986 - val_mean_squared_error: 0.0177
Epoch 19/50
Epoch 00018: val_loss did not improve
1s - loss: 0.0146 - acc: 0.4626 - mean_squared_error: 0.0146 - val_loss: 0.
```

```
0177 - val_acc: 0.6986 - val_mean_squared_error: 0.0177
Epoch 20/50
Epoch 00019: val_loss improved from 0.01765 to 0.01664, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0144 - acc: 0.4556 - mean_squared_error: 0.0144 - val_loss: 0.
0166 - val_acc: 0.6986 - val_mean_squared_error: 0.0166
Epoch 21/50
Epoch 00020: val_loss improved from 0.01664 to 0.01632, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0140 - acc: 0.4655 - mean_squared_error: 0.0140 - val_loss: 0.
0163 - val_acc: 0.6986 - val_mean_squared_error: 0.0163
Epoch 22/50
Epoch 00021: val_loss improved from 0.01632 to 0.01557, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0136 - acc: 0.4614 - mean_squared_error: 0.0136 - val_loss: 0.
0156 - val_acc: 0.6963 - val_mean_squared_error: 0.0156
Epoch 23/50
Epoch 00022: val_loss improved from 0.01557 to 0.01506, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0136 - acc: 0.4825 - mean_squared_error: 0.0136 - val_loss: 0.
0151 - val_acc: 0.6963 - val_mean_squared_error: 0.0151
Epoch 24/50
Epoch 00023: val_loss improved from 0.01506 to 0.01445, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0133 - acc: 0.4731 - mean_squared_error: 0.0133 - val_loss: 0.
0144 - val_acc: 0.6963 - val_mean_squared_error: 0.0144
Epoch 25/50
Epoch 00024: val_loss did not improve
1s - loss: 0.0129 - acc: 0.5146 - mean_squared_error: 0.0129 - val_loss: 0.
0147 - val_acc: 0.6963 - val_mean_squared_error: 0.0147
Epoch 26/50
Epoch 00025: val_loss did not improve
1s - loss: 0.0130 - acc: 0.4749 - mean_squared_error: 0.0130 - val_loss: 0.
0145 - val_acc: 0.6963 - val_mean_squared_error: 0.0145
Epoch 27/50
Epoch 00026: val_loss improved from 0.01445 to 0.01394, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0125 - acc: 0.5093 - mean_squared_error: 0.0125 - val_loss: 0.
0139 - val_acc: 0.6963 - val_mean_squared_error: 0.0139
Epoch 28/50
Epoch 00027: val_loss improved from 0.01394 to 0.01328, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0124 - acc: 0.4988 - mean_squared_error: 0.0124 - val_loss: 0.
0133 - val_acc: 0.6963 - val_mean_squared_error: 0.0133
Epoch 29/50
Epoch 00028: val_loss did not improve
1s - loss: 0.0123 - acc: 0.5058 - mean_squared_error: 0.0123 - val_loss: 0.
0138 - val_acc: 0.6963 - val_mean_squared_error: 0.0138
Epoch 30/50
Epoch 00029: val_loss improved from 0.01328 to 0.01308, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0122 - acc: 0.5000 - mean_squared_error: 0.0122 - val_loss: 0.
0131 - val_acc: 0.6963 - val_mean_squared_error: 0.0131
Epoch 31/50
Epoch 00030: val_loss improved from 0.01308 to 0.01297, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0121 - acc: 0.5088 - mean_squared_error: 0.0121 - val_loss: 0.
```

```
0130 - val_acc: 0.6963 - val_mean_squared_error: 0.0130
Epoch 32/50
Epoch 00031: val_loss improved from 0.01297 to 0.01205, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0119 - acc: 0.5041 - mean_squared_error: 0.0119 - val_loss: 0.
0121 - val_acc: 0.6963 - val_mean_squared_error: 0.0121
Epoch 33/50
Epoch 00032: val_loss did not improve
1s - loss: 0.0118 - acc: 0.4877 - mean_squared_error: 0.0118 - val_loss: 0.
0124 - val_acc: 0.6963 - val_mean_squared_error: 0.0124
Epoch 34/50
Epoch 00033: val_loss did not improve
1s - loss: 0.0116 - acc: 0.5076 - mean_squared_error: 0.0116 - val_loss: 0.
0125 - val_acc: 0.6963 - val_mean_squared_error: 0.0125
Epoch 35/50
Epoch 00034: val_loss did not improve
1s - loss: 0.0116 - acc: 0.5321 - mean_squared_error: 0.0116 - val_loss: 0.
0125 - val_acc: 0.6963 - val_mean_squared_error: 0.0125
Epoch 36/50
Epoch 00035: val_loss improved from 0.01205 to 0.01152, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0115 - acc: 0.5152 - mean_squared_error: 0.0115 - val_loss: 0.
0115 - val_acc: 0.6963 - val_mean_squared_error: 0.0115
Epoch 37/50
Epoch 00036: val_loss did not improve
1s - loss: 0.0115 - acc: 0.5169 - mean_squared_error: 0.0115 - val_loss: 0.
0122 - val_acc: 0.6963 - val_mean_squared_error: 0.0122
Epoch 38/50
Epoch 00037: val_loss did not improve
1s - loss: 0.0113 - acc: 0.5076 - mean_squared_error: 0.0113 - val_loss: 0.
0116 - val_acc: 0.6963 - val_mean_squared_error: 0.0116
Epoch 39/50
Epoch 00038: val_loss improved from 0.01152 to 0.01124, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0112 - acc: 0.5175 - mean_squared_error: 0.0112 - val_loss: 0.
0112 - val_acc: 0.6963 - val_mean_squared_error: 0.0112
Epoch 40/50
Epoch 00039: val_loss improved from 0.01124 to 0.01114, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0110 - acc: 0.5286 - mean_squared_error: 0.0110 - val_loss: 0.
0111 - val_acc: 0.6963 - val_mean_squared_error: 0.0111
Epoch 41/50
Epoch 00040: val_loss improved from 0.01114 to 0.01110, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0110 - acc: 0.5386 - mean_squared_error: 0.0110 - val_loss: 0.
0111 - val_acc: 0.6963 - val_mean_squared_error: 0.0111
Epoch 42/50
Epoch 00041: val_loss did not improve
1s - loss: 0.0109 - acc: 0.5251 - mean_squared_error: 0.0109 - val_loss: 0.
0111 - val_acc: 0.6963 - val_mean_squared_error: 0.0111
Epoch 43/50
Epoch 00042: val_loss improved from 0.01110 to 0.01096, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0108 - acc: 0.5380 - mean_squared_error: 0.0108 - val_loss: 0.
0110 - val_acc: 0.6963 - val_mean_squared_error: 0.0110
Epoch 44/50
Epoch 00043: val_loss improved from 0.01096 to 0.01089, saving model to ./m
```

```
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0107 - acc: 0.5339 - mean_squared_error: 0.0107 - val_loss: 0.
0109 - val_acc: 0.6963 - val_mean_squared_error: 0.0109
Epoch 45/50
Epoch 00044: val_loss improved from 0.01089 to 0.01086, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0107 - acc: 0.5234 - mean_squared_error: 0.0107 - val_loss: 0.
0109 - val_acc: 0.6963 - val_mean_squared_error: 0.0109
Epoch 46/50
Epoch 00045: val_loss did not improve
1s - loss: 0.0106 - acc: 0.5321 - mean_squared_error: 0.0106 - val_loss: 0.
0111 - val_acc: 0.6963 - val_mean_squared_error: 0.0111
Epoch 47/50
Epoch 00046: val_loss did not improve
1s - loss: 0.0105 - acc: 0.5333 - mean_squared_error: 0.0105 - val_loss: 0.
0109 - val_acc: 0.6963 - val_mean_squared_error: 0.0109
Epoch 48/50
Epoch 00047: val_loss did not improve
1s - loss: 0.0103 - acc: 0.5228 - mean_squared_error: 0.0103 - val_loss: 0.
0109 - val_acc: 0.6963 - val_mean_squared_error: 0.0109
Epoch 49/50
Epoch 00048: val_loss improved from 0.01086 to 0.01055, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0105 - acc: 0.5386 - mean_squared_error: 0.0105 - val_loss: 0.
0106 - val_acc: 0.6963 - val_mean_squared_error: 0.0106
Epoch 50/50
Epoch 00049: val_loss improved from 0.01055 to 0.01042, saving model to ./m
odel/SGD_model.weights.best.hdf5
1s - loss: 0.0104 - acc: 0.5187 - mean_squared_error: 0.0104 - val_loss: 0.
0104 - val_acc: 0.6963 - val_mean_squared_error: 0.0104



  Running model: dropout_base_model w/opt: RMSprop
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.06051, saving model to ./mode
l/RMSprop_model.weights.best.hdf5
3s - loss: 0.1892 - acc: 0.4019 - mean_squared_error: 0.1892 - val_loss: 0.
0605 - val_acc: 0.6963 - val_mean_squared_error: 0.0605
Epoch 2/50
Epoch 00001: val_loss improved from 0.06051 to 0.01994, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0223 - acc: 0.4609 - mean_squared_error: 0.0223 - val_loss: 0.
0199 - val_acc: 0.6963 - val_mean_squared_error: 0.0199
Epoch 3/50
Epoch 00002: val_loss improved from 0.01994 to 0.01546, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0134 - acc: 0.5491 - mean_squared_error: 0.0134 - val_loss: 0.
0155 - val_acc: 0.6963 - val_mean_squared_error: 0.0155
Epoch 4/50
Epoch 00003: val_loss improved from 0.01546 to 0.00537, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0092 - acc: 0.5958 - mean_squared_error: 0.0092 - val_loss: 0.
0054 - val_acc: 0.6963 - val_mean_squared_error: 0.0054
Epoch 5/50
Epoch 00004: val_loss did not improve
```

```
1s - loss: 0.0077 - acc: 0.6355 - mean_squared_error: 0.0077 - val_loss: 0.
0081 - val_acc: 0.6963 - val_mean_squared_error: 0.0081
Epoch 6/50
Epoch 00005: val_loss improved from 0.00537 to 0.00511, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0067 - acc: 0.6560 - mean_squared_error: 0.0067 - val_loss: 0.
0051 - val_acc: 0.6963 - val_mean_squared_error: 0.0051
Epoch 7/50
Epoch 00006: val_loss did not improve
1s - loss: 0.0058 - acc: 0.6624 - mean_squared_error: 0.0058 - val_loss: 0.
0054 - val_acc: 0.6986 - val_mean_squared_error: 0.0054
Epoch 8/50
Epoch 00007: val_loss improved from 0.00511 to 0.00442, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0058 - acc: 0.6840 - mean_squared_error: 0.0058 - val_loss: 0.
0044 - val_acc: 0.6963 - val_mean_squared_error: 0.0044
Epoch 9/50
Epoch 00008: val_loss improved from 0.00442 to 0.00325, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0052 - acc: 0.6869 - mean_squared_error: 0.0052 - val_loss: 0.
0033 - val_acc: 0.6986 - val_mean_squared_error: 0.0033
Epoch 10/50
Epoch 00009: val_loss improved from 0.00325 to 0.00313, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0047 - acc: 0.6793 - mean_squared_error: 0.0047 - val_loss: 0.
0031 - val_acc: 0.7103 - val_mean_squared_error: 0.0031
Epoch 11/50
Epoch 00010: val_loss improved from 0.00313 to 0.00295, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0044 - acc: 0.7050 - mean_squared_error: 0.0044 - val_loss: 0.
0030 - val_acc: 0.7103 - val_mean_squared_error: 0.0030
Epoch 12/50
Epoch 00011: val_loss did not improve
1s - loss: 0.0041 - acc: 0.6992 - mean_squared_error: 0.0041 - val_loss: 0.
0042 - val_acc: 0.7103 - val_mean_squared_error: 0.0042
Epoch 13/50
Epoch 00012: val_loss did not improve
1s - loss: 0.0038 - acc: 0.7109 - mean_squared_error: 0.0038 - val_loss: 0.
0044 - val_acc: 0.7196 - val_mean_squared_error: 0.0044
Epoch 14/50
Epoch 00013: val_loss improved from 0.00295 to 0.00215, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0037 - acc: 0.7097 - mean_squared_error: 0.0037 - val_loss: 0.
0022 - val_acc: 0.7150 - val_mean_squared_error: 0.0022
Epoch 15/50
Epoch 00014: val_loss did not improve
1s - loss: 0.0035 - acc: 0.7120 - mean_squared_error: 0.0035 - val_loss: 0.
0022 - val_acc: 0.7150 - val_mean_squared_error: 0.0022
Epoch 16/50
Epoch 00015: val_loss did not improve
1s - loss: 0.0033 - acc: 0.7091 - mean_squared_error: 0.0033 - val_loss: 0.
0028 - val_acc: 0.7103 - val_mean_squared_error: 0.0028
Epoch 17/50
Epoch 00016: val_loss did not improve
1s - loss: 0.0031 - acc: 0.7284 - mean_squared_error: 0.0031 - val_loss: 0.
0041 - val_acc: 0.7103 - val_mean_squared_error: 0.0041
Epoch 18/50
```

```
Epoch 00017: val_loss did not improve
1s - loss: 0.0029 - acc: 0.7044 - mean_squared_error: 0.0029 - val_loss: 0.
0037 - val_acc: 0.7103 - val_mean_squared_error: 0.0037
Epoch 19/50
Epoch 00018: val_loss improved from 0.00215 to 0.00173, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0027 - acc: 0.7301 - mean_squared_error: 0.0027 - val_loss: 0.
0017 - val_acc: 0.7220 - val_mean_squared_error: 0.0017
Epoch 20/50
Epoch 00019: val_loss did not improve
1s - loss: 0.0025 - acc: 0.7243 - mean_squared_error: 0.0025 - val_loss: 0.
0023 - val_acc: 0.7173 - val_mean_squared_error: 0.0023
Epoch 21/50
Epoch 00020: val_loss did not improve
1s - loss: 0.0025 - acc: 0.7249 - mean_squared_error: 0.0025 - val_loss: 0.
0020 - val_acc: 0.7243 - val_mean_squared_error: 0.0020
Epoch 22/50
Epoch 00021: val_loss did not improve
1s - loss: 0.0023 - acc: 0.7173 - mean_squared_error: 0.0023 - val_loss: 0.
0021 - val_acc: 0.7103 - val_mean_squared_error: 0.0021
Epoch 23/50
Epoch 00022: val_loss did not improve
1s - loss: 0.0023 - acc: 0.7190 - mean_squared_error: 0.0023 - val_loss: 0.
0020 - val_acc: 0.7150 - val_mean_squared_error: 0.0020
Epoch 24/50
Epoch 00023: val_loss improved from 0.00173 to 0.00157, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0023 - acc: 0.7307 - mean_squared_error: 0.0023 - val_loss: 0.
0016 - val_acc: 0.7266 - val_mean_squared_error: 0.0016
Epoch 25/50
Epoch 00024: val_loss did not improve
1s - loss: 0.0022 - acc: 0.7377 - mean_squared_error: 0.0022 - val_loss: 0.
0017 - val_acc: 0.7150 - val_mean_squared_error: 0.0017
Epoch 26/50
Epoch 00025: val_loss did not improve
1s - loss: 0.0021 - acc: 0.7301 - mean_squared_error: 0.0021 - val_loss: 0.
0018 - val_acc: 0.7220 - val_mean_squared_error: 0.0018
Epoch 27/50
Epoch 00026: val_loss improved from 0.00157 to 0.00149, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0021 - acc: 0.7383 - mean_squared_error: 0.0021 - val_loss: 0.
0015 - val_acc: 0.7126 - val_mean_squared_error: 0.0015
Epoch 28/50
Epoch 00027: val_loss did not improve
1s - loss: 0.0020 - acc: 0.7354 - mean_squared_error: 0.0020 - val_loss: 0.
0017 - val_acc: 0.7196 - val_mean_squared_error: 0.0017
Epoch 29/50
Epoch 00028: val_loss did not improve
1s - loss: 0.0019 - acc: 0.7307 - mean_squared_error: 0.0019 - val_loss: 0.
0016 - val_acc: 0.7243 - val_mean_squared_error: 0.0016
Epoch 30/50
Epoch 00029: val_loss improved from 0.00149 to 0.00146, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0019 - acc: 0.7465 - mean_squared_error: 0.0019 - val_loss: 0.
0015 - val_acc: 0.7220 - val_mean_squared_error: 0.0015
Epoch 31/50
Epoch 00030: val_loss did not improve
```

```
1s - loss: 0.0019 - acc: 0.7360 - mean_squared_error: 0.0019 - val_loss: 0.
0015 - val_acc: 0.7383 - val_mean_squared_error: 0.0015
Epoch 32/50
Epoch 00031: val_loss did not improve
1s - loss: 0.0018 - acc: 0.7442 - mean_squared_error: 0.0018 - val_loss: 0.
0017 - val_acc: 0.7290 - val_mean_squared_error: 0.0017
Epoch 33/50
Epoch 00032: val_loss did not improve
1s - loss: 0.0018 - acc: 0.7383 - mean_squared_error: 0.0018 - val_loss: 0.
0020 - val_acc: 0.7150 - val_mean_squared_error: 0.0020
Epoch 34/50
Epoch 00033: val_loss improved from 0.00146 to 0.00140, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0017 - acc: 0.7482 - mean_squared_error: 0.0017 - val_loss: 0.
0014 - val_acc: 0.7453 - val_mean_squared_error: 0.0014
Epoch 35/50
Epoch 00034: val_loss did not improve
1s - loss: 0.0017 - acc: 0.7389 - mean_squared_error: 0.0017 - val_loss: 0.
0015 - val_acc: 0.7617 - val_mean_squared_error: 0.0015
Epoch 36/50
Epoch 00035: val_loss improved from 0.00140 to 0.00132, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0016 - acc: 0.7634 - mean_squared_error: 0.0016 - val_loss: 0.
0013 - val_acc: 0.7593 - val_mean_squared_error: 0.0013
Epoch 37/50
Epoch 00036: val_loss improved from 0.00132 to 0.00129, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0015 - acc: 0.7558 - mean_squared_error: 0.0015 - val_loss: 0.
0013 - val_acc: 0.7477 - val_mean_squared_error: 0.0013
Epoch 38/50
Epoch 00037: val_loss did not improve
1s - loss: 0.0015 - acc: 0.7652 - mean_squared_error: 0.0015 - val_loss: 0.
0015 - val_acc: 0.7664 - val_mean_squared_error: 0.0015
Epoch 39/50
Epoch 00038: val_loss did not improve
1s - loss: 0.0016 - acc: 0.7681 - mean_squared_error: 0.0016 - val_loss: 0.
0018 - val_acc: 0.7547 - val_mean_squared_error: 0.0018
Epoch 40/50
Epoch 00039: val_loss improved from 0.00129 to 0.00128, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0015 - acc: 0.7634 - mean_squared_error: 0.0015 - val_loss: 0.
0013 - val_acc: 0.7664 - val_mean_squared_error: 0.0013
Epoch 41/50
Epoch 00040: val_loss did not improve
1s - loss: 0.0015 - acc: 0.7716 - mean_squared_error: 0.0015 - val_loss: 0.
0013 - val_acc: 0.7710 - val_mean_squared_error: 0.0013
Epoch 42/50
Epoch 00041: val_loss did not improve
1s - loss: 0.0014 - acc: 0.7617 - mean_squared_error: 0.0014 - val_loss: 0.
0014 - val_acc: 0.7664 - val_mean_squared_error: 0.0014
Epoch 43/50
Epoch 00042: val_loss did not improve
1s - loss: 0.0014 - acc: 0.7605 - mean_squared_error: 0.0014 - val_loss: 0.
0019 - val_acc: 0.7734 - val_mean_squared_error: 0.0019
Epoch 44/50
Epoch 00043: val_loss improved from 0.00128 to 0.00119, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
```

```
1s - loss: 0.0014 - acc: 0.7810 - mean_squared_error: 0.0014 - val_loss: 0.
0012 - val_acc: 0.7640 - val_mean_squared_error: 0.0012
Epoch 45/50
Epoch 00044: val_loss did not improve
1s - loss: 0.0013 - acc: 0.7827 - mean_squared_error: 0.0013 - val_loss: 0.
0015 - val_acc: 0.7710 - val_mean_squared_error: 0.0015
Epoch 46/50
Epoch 00045: val_loss did not improve
1s - loss: 0.0013 - acc: 0.7921 - mean_squared_error: 0.0013 - val_loss: 0.
0013 - val_acc: 0.7804 - val_mean_squared_error: 0.0013
Epoch 47/50
Epoch 00046: val_loss did not improve
1s - loss: 0.0013 - acc: 0.7769 - mean_squared_error: 0.0013 - val_loss: 0.
0017 - val_acc: 0.7850 - val_mean_squared_error: 0.0017
Epoch 48/50
Epoch 00047: val_loss did not improve
1s - loss: 0.0013 - acc: 0.7827 - mean_squared_error: 0.0013 - val_loss: 0.
0014 - val_acc: 0.7710 - val_mean_squared_error: 0.0014
Epoch 49/50
Epoch 00048: val_loss improved from 0.00119 to 0.00117, saving model to ./m
odel/RMSprop_model.weights.best.hdf5
1s - loss: 0.0013 - acc: 0.7973 - mean_squared_error: 0.0013 - val_loss: 0.
0012 - val_acc: 0.7967 - val_mean_squared_error: 0.0012
Epoch 50/50
Epoch 00049: val_loss did not improve
1s - loss: 0.0012 - acc: 0.7862 - mean_squared_error: 0.0012 - val_loss: 0.
0012 - val_acc: 0.7827 - val_mean_squared_error: 0.0012



  Running model: dropout_base_model w/opt: Adagrad
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.03809, saving model to ./mode
l/Adagrad_model.weights.best.hdf5
3s - loss: 1.3187 - acc: 0.3236 - mean_squared_error: 1.3187 - val_loss: 0.
0381 - val_acc: 0.6963 - val_mean_squared_error: 0.0381
Epoch 2/50
Epoch 00001: val_loss improved from 0.03809 to 0.03041, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0169 - acc: 0.4626 - mean_squared_error: 0.0169 - val_loss: 0.
0304 - val_acc: 0.6963 - val_mean_squared_error: 0.0304
Epoch 3/50
Epoch 00002: val_loss improved from 0.03041 to 0.02420, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0141 - acc: 0.4690 - mean_squared_error: 0.0141 - val_loss: 0.
0242 - val_acc: 0.6963 - val_mean_squared_error: 0.0242
Epoch 4/50
Epoch 00003: val_loss improved from 0.02420 to 0.02176, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0128 - acc: 0.5058 - mean_squared_error: 0.0128 - val_loss: 0.
0218 - val_acc: 0.6963 - val_mean_squared_error: 0.0218
Epoch 5/50
Epoch 00004: val_loss did not improve
1s - loss: 0.0122 - acc: 0.5222 - mean_squared_error: 0.0122 - val_loss: 0.
0253 - val_acc: 0.6963 - val_mean_squared_error: 0.0253
Epoch 6/50
```

```
Epoch 00005: val_loss did not improve
1s - loss: 0.0117 - acc: 0.5327 - mean_squared_error: 0.0117 - val_loss: 0.
0280 - val_acc: 0.6963 - val_mean_squared_error: 0.0280
Epoch 7/50
Epoch 00006: val_loss did not improve
1s - loss: 0.0111 - acc: 0.5257 - mean_squared_error: 0.0111 - val_loss: 0.
0243 - val_acc: 0.6963 - val_mean_squared_error: 0.0243
Epoch 8/50
Epoch 00007: val_loss did not improve
1s - loss: 0.0108 - acc: 0.5397 - mean_squared_error: 0.0108 - val_loss: 0.
0253 - val_acc: 0.6963 - val_mean_squared_error: 0.0253
Epoch 9/50
Epoch 00008: val_loss did not improve
1s - loss: 0.0106 - acc: 0.5695 - mean_squared_error: 0.0106 - val_loss: 0.
0235 - val_acc: 0.6963 - val_mean_squared_error: 0.0235
Epoch 10/50
Epoch 00009: val_loss improved from 0.02176 to 0.01870, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0102 - acc: 0.5496 - mean_squared_error: 0.0102 - val_loss: 0.
0187 - val_acc: 0.6963 - val_mean_squared_error: 0.0187
Epoch 11/50
Epoch 00010: val_loss did not improve
1s - loss: 0.0101 - acc: 0.5794 - mean_squared_error: 0.0101 - val_loss: 0.
0247 - val_acc: 0.6963 - val_mean_squared_error: 0.0247
Epoch 12/50
Epoch 00011: val_loss did not improve
1s - loss: 0.0099 - acc: 0.5952 - mean_squared_error: 0.0099 - val_loss: 0.
0241 - val_acc: 0.6963 - val_mean_squared_error: 0.0241
Epoch 13/50
Epoch 00012: val_loss did not improve
1s - loss: 0.0097 - acc: 0.5724 - mean_squared_error: 0.0097 - val_loss: 0.
0215 - val_acc: 0.6963 - val_mean_squared_error: 0.0215
Epoch 14/50
Epoch 00013: val_loss did not improve
1s - loss: 0.0098 - acc: 0.5800 - mean_squared_error: 0.0098 - val_loss: 0.
0213 - val_acc: 0.6963 - val_mean_squared_error: 0.0213
Epoch 15/50
Epoch 00014: val_loss did not improve
1s - loss: 0.0095 - acc: 0.5794 - mean_squared_error: 0.0095 - val_loss: 0.
0213 - val_acc: 0.6963 - val_mean_squared_error: 0.0213
Epoch 16/50
Epoch 00015: val_loss did not improve
1s - loss: 0.0092 - acc: 0.5754 - mean_squared_error: 0.0092 - val_loss: 0.
0233 - val_acc: 0.6963 - val_mean_squared_error: 0.0233
Epoch 17/50
Epoch 00016: val_loss improved from 0.01870 to 0.01725, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0092 - acc: 0.5882 - mean_squared_error: 0.0092 - val_loss: 0.
0172 - val_acc: 0.6963 - val_mean_squared_error: 0.0172
Epoch 18/50
Epoch 00017: val_loss did not improve
1s - loss: 0.0089 - acc: 0.6063 - mean_squared_error: 0.0089 - val_loss: 0.
0191 - val_acc: 0.6963 - val_mean_squared_error: 0.0191
Epoch 19/50
Epoch 00018: val_loss did not improve
1s - loss: 0.0091 - acc: 0.6110 - mean_squared_error: 0.0091 - val_loss: 0.
0205 - val_acc: 0.6963 - val_mean_squared_error: 0.0205
```

```
Epoch 20/50
Epoch 00019: val_loss did not improve
1s - loss: 0.0089 - acc: 0.5853 - mean_squared_error: 0.0089 - val_loss: 0.
0218 - val_acc: 0.6963 - val_mean_squared_error: 0.0218
Epoch 21/50
Epoch 00020: val_loss did not improve
1s - loss: 0.0089 - acc: 0.5987 - mean_squared_error: 0.0089 - val_loss: 0.
0237 - val_acc: 0.6963 - val_mean_squared_error: 0.0237
Epoch 22/50
Epoch 00021: val_loss did not improve
1s - loss: 0.0088 - acc: 0.6192 - mean_squared_error: 0.0088 - val_loss: 0.
0222 - val_acc: 0.6963 - val_mean_squared_error: 0.0222
Epoch 23/50
Epoch 00022: val_loss did not improve
1s - loss: 0.0087 - acc: 0.6051 - mean_squared_error: 0.0087 - val_loss: 0.
0176 - val_acc: 0.6963 - val_mean_squared_error: 0.0176
Epoch 24/50
Epoch 00023: val_loss did not improve
1s - loss: 0.0089 - acc: 0.6151 - mean_squared_error: 0.0089 - val_loss: 0.
0175 - val_acc: 0.6963 - val_mean_squared_error: 0.0175
Epoch 25/50
Epoch 00024: val_loss did not improve
1s - loss: 0.0087 - acc: 0.6268 - mean_squared_error: 0.0087 - val_loss: 0.
0192 - val_acc: 0.6963 - val_mean_squared_error: 0.0192
Epoch 26/50
Epoch 00025: val_loss did not improve
1s - loss: 0.0088 - acc: 0.6238 - mean_squared_error: 0.0088 - val_loss: 0.
0182 - val_acc: 0.6963 - val_mean_squared_error: 0.0182
Epoch 27/50
Epoch 00026: val_loss did not improve
1s - loss: 0.0086 - acc: 0.6262 - mean_squared_error: 0.0086 - val_loss: 0.
0183 - val_acc: 0.6963 - val_mean_squared_error: 0.0183
Epoch 28/50
Epoch 00027: val_loss improved from 0.01725 to 0.01683, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0086 - acc: 0.6268 - mean_squared_error: 0.0086 - val_loss: 0.
0168 - val_acc: 0.6963 - val_mean_squared_error: 0.0168
Epoch 29/50
Epoch 00028: val_loss improved from 0.01683 to 0.01537, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0086 - acc: 0.6192 - mean_squared_error: 0.0086 - val_loss: 0.
0154 - val_acc: 0.6963 - val_mean_squared_error: 0.0154
Epoch 30/50
Epoch 00029: val_loss did not improve
1s - loss: 0.0084 - acc: 0.6168 - mean_squared_error: 0.0084 - val_loss: 0.
0162 - val_acc: 0.6963 - val_mean_squared_error: 0.0162
Epoch 31/50
Epoch 00030: val_loss improved from 0.01537 to 0.01503, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0082 - acc: 0.6379 - mean_squared_error: 0.0082 - val_loss: 0.
0150 - val_acc: 0.6963 - val_mean_squared_error: 0.0150
Epoch 32/50
Epoch 00031: val_loss did not improve
1s - loss: 0.0082 - acc: 0.6203 - mean_squared_error: 0.0082 - val_loss: 0.
0167 - val_acc: 0.6963 - val_mean_squared_error: 0.0167
Epoch 33/50
Epoch 00032: val_loss did not improve
```

```
1s - loss: 0.0081 - acc: 0.6262 - mean_squared_error: 0.0081 - val_loss: 0.
0173 - val_acc: 0.6963 - val_mean_squared_error: 0.0173
Epoch 34/50
Epoch 00033: val_loss did not improve
1s - loss: 0.0082 - acc: 0.6273 - mean_squared_error: 0.0082 - val_loss: 0.
0156 - val_acc: 0.6963 - val_mean_squared_error: 0.0156
Epoch 35/50
Epoch 00034: val_loss did not improve
1s - loss: 0.0082 - acc: 0.6285 - mean_squared_error: 0.0082 - val_loss: 0.
0185 - val_acc: 0.6963 - val_mean_squared_error: 0.0185
Epoch 36/50
Epoch 00035: val_loss did not improve
1s - loss: 0.0081 - acc: 0.6373 - mean_squared_error: 0.0081 - val_loss: 0.
0163 - val_acc: 0.6963 - val_mean_squared_error: 0.0163
Epoch 37/50
Epoch 00036: val_loss did not improve
1s - loss: 0.0080 - acc: 0.6519 - mean_squared_error: 0.0080 - val_loss: 0.
0164 - val_acc: 0.6963 - val_mean_squared_error: 0.0164
Epoch 38/50
Epoch 00037: val_loss improved from 0.01503 to 0.01263, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0078 - acc: 0.6414 - mean_squared_error: 0.0078 - val_loss: 0.
0126 - val_acc: 0.6963 - val_mean_squared_error: 0.0126
Epoch 39/50
Epoch 00038: val_loss did not improve
1s - loss: 0.0077 - acc: 0.6367 - mean_squared_error: 0.0077 - val_loss: 0.
0150 - val_acc: 0.6963 - val_mean_squared_error: 0.0150
Epoch 40/50
Epoch 00039: val_loss did not improve
1s - loss: 0.0078 - acc: 0.6542 - mean_squared_error: 0.0078 - val_loss: 0.
0147 - val_acc: 0.6963 - val_mean_squared_error: 0.0147
Epoch 41/50
Epoch 00040: val_loss did not improve
1s - loss: 0.0077 - acc: 0.6379 - mean_squared_error: 0.0077 - val_loss: 0.
0176 - val_acc: 0.6963 - val_mean_squared_error: 0.0176
Epoch 42/50
Epoch 00041: val_loss did not improve
1s - loss: 0.0076 - acc: 0.6484 - mean_squared_error: 0.0076 - val_loss: 0.
0138 - val_acc: 0.6986 - val_mean_squared_error: 0.0138
Epoch 43/50
Epoch 00042: val_loss did not improve
1s - loss: 0.0078 - acc: 0.6408 - mean_squared_error: 0.0078 - val_loss: 0.
0139 - val_acc: 0.6963 - val_mean_squared_error: 0.0139
Epoch 44/50
Epoch 00043: val_loss did not improve
1s - loss: 0.0074 - acc: 0.6565 - mean_squared_error: 0.0074 - val_loss: 0.
0149 - val_acc: 0.6986 - val_mean_squared_error: 0.0149
Epoch 45/50
Epoch 00044: val_loss improved from 0.01263 to 0.01204, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0076 - acc: 0.6460 - mean_squared_error: 0.0076 - val_loss: 0.
0120 - val_acc: 0.6963 - val_mean_squared_error: 0.0120
Epoch 46/50
Epoch 00045: val_loss did not improve
1s - loss: 0.0076 - acc: 0.6641 - mean_squared_error: 0.0076 - val_loss: 0.
0150 - val_acc: 0.6986 - val_mean_squared_error: 0.0150
Epoch 47/50
```

```
Epoch 00046: val_loss did not improve
1s - loss: 0.0073 - acc: 0.6332 - mean_squared_error: 0.0073 - val_loss: 0.
0143 - val_acc: 0.6963 - val_mean_squared_error: 0.0143
Epoch 48/50
Epoch 00047: val_loss did not improve
1s - loss: 0.0072 - acc: 0.6507 - mean_squared_error: 0.0072 - val_loss: 0.
0129 - val_acc: 0.6963 - val_mean_squared_error: 0.0129
Epoch 49/50
Epoch 00048: val_loss did not improve
1s - loss: 0.0069 - acc: 0.6542 - mean_squared_error: 0.0069 - val_loss: 0.
0129 - val_acc: 0.6986 - val_mean_squared_error: 0.0129
Epoch 50/50
Epoch 00049: val_loss improved from 0.01204 to 0.01175, saving model to ./m
odel/Adagrad_model.weights.best.hdf5
1s - loss: 0.0072 - acc: 0.6595 - mean_squared_error: 0.0072 - val_loss: 0.
0117 - val_acc: 0.6986 - val_mean_squared_error: 0.0117



  Running model: dropout_base_model w/opt: Adadelta
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.01602, saving model to ./mode
l/Adadelta_model.weights.best.hdf5
3s - loss: 0.0240 - acc: 0.4001 - mean_squared_error: 0.0240 - val_loss: 0.
0160 - val_acc: 0.6963 - val_mean_squared_error: 0.0160
Epoch 2/50
Epoch 00001: val_loss did not improve
1s - loss: 0.0129 - acc: 0.5327 - mean_squared_error: 0.0129 - val_loss: 0.
0190 - val_acc: 0.6963 - val_mean_squared_error: 0.0190
Epoch 3/50
Epoch 00002: val_loss improved from 0.01602 to 0.00796, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0105 - acc: 0.5444 - mean_squared_error: 0.0105 - val_loss: 0.
0080 - val_acc: 0.6963 - val_mean_squared_error: 0.0080
Epoch 4/50
Epoch 00003: val_loss did not improve
1s - loss: 0.0091 - acc: 0.5847 - mean_squared_error: 0.0091 - val_loss: 0.
0107 - val_acc: 0.6963 - val_mean_squared_error: 0.0107
Epoch 5/50
Epoch 00004: val_loss improved from 0.00796 to 0.00660, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0084 - acc: 0.5940 - mean_squared_error: 0.0084 - val_loss: 0.
0066 - val_acc: 0.6963 - val_mean_squared_error: 0.0066
Epoch 6/50
Epoch 00005: val_loss did not improve
1s - loss: 0.0079 - acc: 0.6063 - mean_squared_error: 0.0079 - val_loss: 0.
0150 - val_acc: 0.6963 - val_mean_squared_error: 0.0150
Epoch 7/50
Epoch 00006: val_loss did not improve
1s - loss: 0.0075 - acc: 0.6203 - mean_squared_error: 0.0075 - val_loss: 0.
0085 - val_acc: 0.6963 - val_mean_squared_error: 0.0085
Epoch 8/50
Epoch 00007: val_loss did not improve
1s - loss: 0.0071 - acc: 0.6355 - mean_squared_error: 0.0071 - val_loss: 0.
0095 - val_acc: 0.6963 - val_mean_squared_error: 0.0095
Epoch 9/50
```

```
Epoch 00008: val_loss did not improve
1s - loss: 0.0066 - acc: 0.6355 - mean_squared_error: 0.0066 - val_loss: 0.
0074 - val_acc: 0.6963 - val_mean_squared_error: 0.0074
Epoch 10/50
Epoch 00009: val_loss did not improve
1s - loss: 0.0065 - acc: 0.6513 - mean_squared_error: 0.0065 - val_loss: 0.
0087 - val_acc: 0.6963 - val_mean_squared_error: 0.0087
Epoch 11/50
Epoch 00010: val_loss improved from 0.00660 to 0.00658, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0062 - acc: 0.6495 - mean_squared_error: 0.0062 - val_loss: 0.
0066 - val_acc: 0.6963 - val_mean_squared_error: 0.0066
Epoch 12/50
Epoch 00011: val_loss did not improve
1s - loss: 0.0062 - acc: 0.6700 - mean_squared_error: 0.0062 - val_loss: 0.
0099 - val_acc: 0.6963 - val_mean_squared_error: 0.0099
Epoch 13/50
Epoch 00012: val_loss did not improve
1s - loss: 0.0060 - acc: 0.6676 - mean_squared_error: 0.0060 - val_loss: 0.
0101 - val_acc: 0.6963 - val_mean_squared_error: 0.0101
Epoch 14/50
Epoch 00013: val_loss did not improve
1s - loss: 0.0058 - acc: 0.6711 - mean_squared_error: 0.0058 - val_loss: 0.
0090 - val_acc: 0.6963 - val_mean_squared_error: 0.0090
Epoch 15/50
Epoch 00014: val_loss did not improve
1s - loss: 0.0056 - acc: 0.6875 - mean_squared_error: 0.0056 - val_loss: 0.
0072 - val_acc: 0.6963 - val_mean_squared_error: 0.0072
Epoch 16/50
Epoch 00015: val_loss did not improve
1s - loss: 0.0055 - acc: 0.6770 - mean_squared_error: 0.0055 - val_loss: 0.
0085 - val_acc: 0.6963 - val_mean_squared_error: 0.0085
Epoch 17/50
Epoch 00016: val_loss did not improve
1s - loss: 0.0055 - acc: 0.6828 - mean_squared_error: 0.0055 - val_loss: 0.
0071 - val_acc: 0.6963 - val_mean_squared_error: 0.0071
Epoch 18/50
Epoch 00017: val_loss improved from 0.00658 to 0.00634, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0054 - acc: 0.6904 - mean_squared_error: 0.0054 - val_loss: 0.
0063 - val_acc: 0.6963 - val_mean_squared_error: 0.0063
Epoch 19/50
Epoch 00018: val_loss improved from 0.00634 to 0.00618, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0052 - acc: 0.6805 - mean_squared_error: 0.0052 - val_loss: 0.
0062 - val_acc: 0.6963 - val_mean_squared_error: 0.0062
Epoch 20/50
Epoch 00019: val_loss improved from 0.00618 to 0.00442, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0051 - acc: 0.6834 - mean_squared_error: 0.0051 - val_loss: 0.
0044 - val_acc: 0.6963 - val_mean_squared_error: 0.0044
Epoch 21/50
Epoch 00020: val_loss did not improve
1s - loss: 0.0051 - acc: 0.6881 - mean_squared_error: 0.0051 - val_loss: 0.
0047 - val_acc: 0.6963 - val_mean_squared_error: 0.0047
Epoch 22/50
Epoch 00021: val_loss did not improve
```

```
1s - loss: 0.0050 - acc: 0.6875 - mean_squared_error: 0.0050 - val_loss: 0.
0073 - val_acc: 0.6963 - val_mean_squared_error: 0.0073
Epoch 23/50
Epoch 00022: val_loss did not improve
1s - loss: 0.0049 - acc: 0.6968 - mean_squared_error: 0.0049 - val_loss: 0.
0047 - val_acc: 0.6963 - val_mean_squared_error: 0.0047
Epoch 24/50
Epoch 00023: val_loss did not improve
1s - loss: 0.0048 - acc: 0.6957 - mean_squared_error: 0.0048 - val_loss: 0.
0059 - val_acc: 0.6963 - val_mean_squared_error: 0.0059
Epoch 25/50
Epoch 00024: val_loss did not improve
1s - loss: 0.0047 - acc: 0.6945 - mean_squared_error: 0.0047 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 26/50
Epoch 00025: val_loss did not improve
1s - loss: 0.0047 - acc: 0.7039 - mean_squared_error: 0.0047 - val_loss: 0.
0046 - val_acc: 0.6963 - val_mean_squared_error: 0.0046
Epoch 27/50
Epoch 00026: val_loss did not improve
1s - loss: 0.0046 - acc: 0.7004 - mean_squared_error: 0.0046 - val_loss: 0.
0046 - val_acc: 0.6963 - val_mean_squared_error: 0.0046
Epoch 28/50
Epoch 00027: val_loss did not improve
1s - loss: 0.0045 - acc: 0.6933 - mean_squared_error: 0.0045 - val_loss: 0.
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 29/50
Epoch 00028: val_loss improved from 0.00442 to 0.00439, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0045 - acc: 0.6992 - mean_squared_error: 0.0045 - val_loss: 0.
0044 - val_acc: 0.6963 - val_mean_squared_error: 0.0044
Epoch 30/50
Epoch 00029: val_loss did not improve
1s - loss: 0.0043 - acc: 0.6957 - mean_squared_error: 0.0043 - val_loss: 0.
0048 - val_acc: 0.6963 - val_mean_squared_error: 0.0048
Epoch 31/50
Epoch 00030: val_loss improved from 0.00439 to 0.00433, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0042 - acc: 0.6957 - mean_squared_error: 0.0042 - val_loss: 0.
0043 - val_acc: 0.6963 - val_mean_squared_error: 0.0043
Epoch 32/50
Epoch 00031: val_loss did not improve
1s - loss: 0.0042 - acc: 0.6951 - mean_squared_error: 0.0042 - val_loss: 0.
0045 - val_acc: 0.6963 - val_mean_squared_error: 0.0045
Epoch 33/50
Epoch 00032: val_loss improved from 0.00433 to 0.00375, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0041 - acc: 0.6998 - mean_squared_error: 0.0041 - val_loss: 0.
0038 - val_acc: 0.6963 - val_mean_squared_error: 0.0038
Epoch 34/50
Epoch 00033: val_loss did not improve
1s - loss: 0.0041 - acc: 0.6974 - mean_squared_error: 0.0041 - val_loss: 0.
0043 - val_acc: 0.6963 - val_mean_squared_error: 0.0043
Epoch 35/50
Epoch 00034: val_loss did not improve
1s - loss: 0.0040 - acc: 0.7015 - mean_squared_error: 0.0040 - val_loss: 0.
0046 - val_acc: 0.6963 - val_mean_squared_error: 0.0046
```

```
Epoch 36/50
Epoch 00035: val_loss did not improve
1s - loss: 0.0039 - acc: 0.7021 - mean_squared_error: 0.0039 - val_loss: 0.
0047 - val_acc: 0.6963 - val_mean_squared_error: 0.0047
Epoch 37/50
Epoch 00036: val_loss did not improve
1s - loss: 0.0039 - acc: 0.7050 - mean_squared_error: 0.0039 - val_loss: 0.
0038 - val_acc: 0.6963 - val_mean_squared_error: 0.0038
Epoch 38/50
Epoch 00037: val_loss did not improve
1s - loss: 0.0038 - acc: 0.7132 - mean_squared_error: 0.0038 - val_loss: 0.
0057 - val_acc: 0.7009 - val_mean_squared_error: 0.0057
Epoch 39/50
Epoch 00038: val_loss did not improve
1s - loss: 0.0038 - acc: 0.7062 - mean_squared_error: 0.0038 - val_loss: 0.
0038 - val_acc: 0.6963 - val_mean_squared_error: 0.0038
Epoch 40/50
Epoch 00039: val_loss improved from 0.00375 to 0.00362, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0037 - acc: 0.6980 - mean_squared_error: 0.0037 - val_loss: 0.
0036 - val_acc: 0.6986 - val_mean_squared_error: 0.0036
Epoch 41/50
Epoch 00040: val_loss did not improve
1s - loss: 0.0036 - acc: 0.7085 - mean_squared_error: 0.0036 - val_loss: 0.
0042 - val_acc: 0.7009 - val_mean_squared_error: 0.0042
Epoch 42/50
Epoch 00041: val_loss improved from 0.00362 to 0.00357, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0036 - acc: 0.6939 - mean_squared_error: 0.0036 - val_loss: 0.
0036 - val_acc: 0.7009 - val_mean_squared_error: 0.0036
Epoch 43/50
Epoch 00042: val_loss improved from 0.00357 to 0.00344, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0035 - acc: 0.7085 - mean_squared_error: 0.0035 - val_loss: 0.
0034 - val_acc: 0.7009 - val_mean_squared_error: 0.0034
Epoch 44/50
Epoch 00043: val_loss did not improve
1s - loss: 0.0034 - acc: 0.6986 - mean_squared_error: 0.0034 - val_loss: 0.
0038 - val_acc: 0.7009 - val_mean_squared_error: 0.0038
Epoch 45/50
Epoch 00044: val_loss improved from 0.00344 to 0.00302, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0034 - acc: 0.7009 - mean_squared_error: 0.0034 - val_loss: 0.
0030 - val_acc: 0.7009 - val_mean_squared_error: 0.0030
Epoch 46/50
Epoch 00045: val_loss improved from 0.00302 to 0.00281, saving model to ./m
odel/Adadelta_model.weights.best.hdf5
1s - loss: 0.0033 - acc: 0.7009 - mean_squared_error: 0.0033 - val_loss: 0.
0028 - val_acc: 0.7009 - val_mean_squared_error: 0.0028
Epoch 47/50
Epoch 00046: val_loss did not improve
1s - loss: 0.0034 - acc: 0.6957 - mean_squared_error: 0.0034 - val_loss: 0.
0028 - val_acc: 0.7009 - val_mean_squared_error: 0.0028
Epoch 48/50
Epoch 00047: val_loss did not improve
1s - loss: 0.0033 - acc: 0.7015 - mean_squared_error: 0.0033 - val_loss: 0.
0029 - val_acc: 0.6986 - val_mean_squared_error: 0.0029
```

```
Epoch 49/50
Epoch 00048: val_loss did not improve
1s - loss: 0.0032 - acc: 0.7004 - mean_squared_error: 0.0032 - val_loss: 0.
0031 - val_acc: 0.7009 - val_mean_squared_error: 0.0031
Epoch 50/50
Epoch 00049: val_loss did not improve
1s - loss: 0.0032 - acc: 0.7097 - mean_squared_error: 0.0032 - val_loss: 0.
0029 - val_acc: 0.7009 - val_mean_squared_error: 0.0029



  Running model: dropout_base_model w/opt: Adam
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.07780, saving model to ./mode
l/Adam_model.weights.best.hdf5
3s - loss: 0.0630 - acc: 0.2845 - mean_squared_error: 0.0630 - val_loss: 0.
0778 - val_acc: 0.6822 - val_mean_squared_error: 0.0778
Epoch 2/50
Epoch 00001: val_loss improved from 0.07780 to 0.05070, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0158 - acc: 0.4322 - mean_squared_error: 0.0158 - val_loss: 0.
0507 - val_acc: 0.6682 - val_mean_squared_error: 0.0507
Epoch 3/50
Epoch 00002: val_loss did not improve
1s - loss: 0.0116 - acc: 0.5111 - mean_squared_error: 0.0116 - val_loss: 0.
0520 - val_acc: 0.6963 - val_mean_squared_error: 0.0520
Epoch 4/50
Epoch 00003: val_loss improved from 0.05070 to 0.04590, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0098 - acc: 0.5432 - mean_squared_error: 0.0098 - val_loss: 0.
0459 - val_acc: 0.6963 - val_mean_squared_error: 0.0459
Epoch 5/50
Epoch 00004: val_loss improved from 0.04590 to 0.03359, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0091 - acc: 0.5724 - mean_squared_error: 0.0091 - val_loss: 0.
0336 - val_acc: 0.6963 - val_mean_squared_error: 0.0336
Epoch 6/50
Epoch 00005: val_loss did not improve
1s - loss: 0.0081 - acc: 0.6016 - mean_squared_error: 0.0081 - val_loss: 0.
0362 - val_acc: 0.6986 - val_mean_squared_error: 0.0362
Epoch 7/50
Epoch 00006: val_loss improved from 0.03359 to 0.02460, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0078 - acc: 0.6121 - mean_squared_error: 0.0078 - val_loss: 0.
0246 - val_acc: 0.6963 - val_mean_squared_error: 0.0246
Epoch 8/50
Epoch 00007: val_loss improved from 0.02460 to 0.02105, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0073 - acc: 0.6414 - mean_squared_error: 0.0073 - val_loss: 0.
0211 - val_acc: 0.6963 - val_mean_squared_error: 0.0211
Epoch 9/50
Epoch 00008: val_loss improved from 0.02105 to 0.01922, saving model to ./m
odel/Adam_model.weights.best.hdf5
2s - loss: 0.0068 - acc: 0.6530 - mean_squared_error: 0.0068 - val_loss: 0.
0192 - val_acc: 0.6986 - val_mean_squared_error: 0.0192
Epoch 10/50
```

```
Epoch 00009: val_loss did not improve
1s - loss: 0.0065 - acc: 0.6454 - mean_squared_error: 0.0065 - val_loss: 0.
0207 - val_acc: 0.6963 - val_mean_squared_error: 0.0207
Epoch 11/50
Epoch 00010: val_loss improved from 0.01922 to 0.01819, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0062 - acc: 0.6647 - mean_squared_error: 0.0062 - val_loss: 0.
0182 - val_acc: 0.6963 - val_mean_squared_error: 0.0182
Epoch 12/50
Epoch 00011: val_loss improved from 0.01819 to 0.01432, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0061 - acc: 0.6612 - mean_squared_error: 0.0061 - val_loss: 0.
0143 - val_acc: 0.6986 - val_mean_squared_error: 0.0143
Epoch 13/50
Epoch 00012: val_loss improved from 0.01432 to 0.01276, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0058 - acc: 0.6817 - mean_squared_error: 0.0058 - val_loss: 0.
0128 - val_acc: 0.6963 - val_mean_squared_error: 0.0128
Epoch 14/50
Epoch 00013: val_loss did not improve
1s - loss: 0.0056 - acc: 0.7033 - mean_squared_error: 0.0056 - val_loss: 0.
0130 - val_acc: 0.6986 - val_mean_squared_error: 0.0130
Epoch 15/50
Epoch 00014: val_loss improved from 0.01276 to 0.01033, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0054 - acc: 0.6974 - mean_squared_error: 0.0054 - val_loss: 0.
0103 - val_acc: 0.6963 - val_mean_squared_error: 0.0103
Epoch 16/50
Epoch 00015: val_loss did not improve
1s - loss: 0.0052 - acc: 0.6957 - mean_squared_error: 0.0052 - val_loss: 0.
0112 - val_acc: 0.6963 - val_mean_squared_error: 0.0112
Epoch 17/50
Epoch 00016: val_loss improved from 0.01033 to 0.00748, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0050 - acc: 0.7056 - mean_squared_error: 0.0050 - val_loss: 0.
0075 - val_acc: 0.6963 - val_mean_squared_error: 0.0075
Epoch 18/50
Epoch 00017: val_loss improved from 0.00748 to 0.00620, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0049 - acc: 0.7091 - mean_squared_error: 0.0049 - val_loss: 0.
0062 - val_acc: 0.7009 - val_mean_squared_error: 0.0062
Epoch 19/50
Epoch 00018: val_loss improved from 0.00620 to 0.00617, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0046 - acc: 0.7120 - mean_squared_error: 0.0046 - val_loss: 0.
0062 - val_acc: 0.6986 - val_mean_squared_error: 0.0062
Epoch 20/50
Epoch 00019: val_loss improved from 0.00617 to 0.00565, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0045 - acc: 0.7091 - mean_squared_error: 0.0045 - val_loss: 0.
0057 - val_acc: 0.7009 - val_mean_squared_error: 0.0057
Epoch 21/50
Epoch 00020: val_loss improved from 0.00565 to 0.00533, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0043 - acc: 0.7009 - mean_squared_error: 0.0043 - val_loss: 0.
0053 - val_acc: 0.7033 - val_mean_squared_error: 0.0053
Epoch 22/50
```

```
Epoch 00021: val_loss improved from 0.00533 to 0.00463, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0042 - acc: 0.7068 - mean_squared_error: 0.0042 - val_loss: 0.
0046 - val_acc: 0.7056 - val_mean_squared_error: 0.0046
Epoch 23/50
Epoch 00022: val_loss improved from 0.00463 to 0.00410, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0041 - acc: 0.7039 - mean_squared_error: 0.0041 - val_loss: 0.
0041 - val_acc: 0.7056 - val_mean_squared_error: 0.0041
Epoch 24/50
Epoch 00023: val_loss improved from 0.00410 to 0.00368, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0039 - acc: 0.7097 - mean_squared_error: 0.0039 - val_loss: 0.
0037 - val_acc: 0.7033 - val_mean_squared_error: 0.0037
Epoch 25/50
Epoch 00024: val_loss improved from 0.00368 to 0.00317, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0038 - acc: 0.7074 - mean_squared_error: 0.0038 - val_loss: 0.
0032 - val_acc: 0.7009 - val_mean_squared_error: 0.0032
Epoch 26/50
Epoch 00025: val_loss did not improve
1s - loss: 0.0037 - acc: 0.7044 - mean_squared_error: 0.0037 - val_loss: 0.
0036 - val_acc: 0.7056 - val_mean_squared_error: 0.0036
Epoch 27/50
Epoch 00026: val_loss improved from 0.00317 to 0.00315, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0036 - acc: 0.7103 - mean_squared_error: 0.0036 - val_loss: 0.
0031 - val_acc: 0.7079 - val_mean_squared_error: 0.0031
Epoch 28/50
Epoch 00027: val_loss improved from 0.00315 to 0.00286, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0035 - acc: 0.7132 - mean_squared_error: 0.0035 - val_loss: 0.
0029 - val_acc: 0.7079 - val_mean_squared_error: 0.0029
Epoch 29/50
Epoch 00028: val_loss improved from 0.00286 to 0.00267, saving model to ./m
odel/Adam_model.weights.best.hdf5
2s - loss: 0.0033 - acc: 0.7044 - mean_squared_error: 0.0033 - val_loss: 0.
0027 - val_acc: 0.7056 - val_mean_squared_error: 0.0027
Epoch 30/50
Epoch 00029: val_loss improved from 0.00267 to 0.00258, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0032 - acc: 0.7126 - mean_squared_error: 0.0032 - val_loss: 0.
0026 - val_acc: 0.7103 - val_mean_squared_error: 0.0026
Epoch 31/50
Epoch 00030: val_loss improved from 0.00258 to 0.00257, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0032 - acc: 0.7068 - mean_squared_error: 0.0032 - val_loss: 0.
0026 - val_acc: 0.7150 - val_mean_squared_error: 0.0026
Epoch 32/50
Epoch 00031: val_loss improved from 0.00257 to 0.00232, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0031 - acc: 0.7214 - mean_squared_error: 0.0031 - val_loss: 0.
0023 - val_acc: 0.7079 - val_mean_squared_error: 0.0023
Epoch 33/50
Epoch 00032: val_loss improved from 0.00232 to 0.00229, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0030 - acc: 0.7202 - mean_squared_error: 0.0030 - val_loss: 0.
```

```
0023 - val_acc: 0.7103 - val_mean_squared_error: 0.0023
Epoch 34/50
Epoch 00033: val_loss did not improve
1s - loss: 0.0029 - acc: 0.7027 - mean_squared_error: 0.0029 - val_loss: 0.
0024 - val_acc: 0.6986 - val_mean_squared_error: 0.0024
Epoch 35/50
Epoch 00034: val_loss did not improve
1s - loss: 0.0029 - acc: 0.7114 - mean_squared_error: 0.0029 - val_loss: 0.
0024 - val_acc: 0.7079 - val_mean_squared_error: 0.0024
Epoch 36/50
Epoch 00035: val_loss improved from 0.00229 to 0.00213, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0028 - acc: 0.7190 - mean_squared_error: 0.0028 - val_loss: 0.
0021 - val_acc: 0.7056 - val_mean_squared_error: 0.0021
Epoch 37/50
Epoch 00036: val_loss did not improve
1s - loss: 0.0027 - acc: 0.7173 - mean_squared_error: 0.0027 - val_loss: 0.
0022 - val_acc: 0.7243 - val_mean_squared_error: 0.0022
Epoch 38/50
Epoch 00037: val_loss improved from 0.00213 to 0.00200, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0026 - acc: 0.7161 - mean_squared_error: 0.0026 - val_loss: 0.
0020 - val_acc: 0.7056 - val_mean_squared_error: 0.0020
Epoch 39/50
Epoch 00038: val_loss did not improve
1s - loss: 0.0026 - acc: 0.7325 - mean_squared_error: 0.0026 - val_loss: 0.
0021 - val_acc: 0.7126 - val_mean_squared_error: 0.0021
Epoch 40/50
Epoch 00039: val_loss improved from 0.00200 to 0.00193, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0025 - acc: 0.7208 - mean_squared_error: 0.0025 - val_loss: 0.
0019 - val_acc: 0.7150 - val_mean_squared_error: 0.0019
Epoch 41/50
Epoch 00040: val_loss improved from 0.00193 to 0.00190, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0025 - acc: 0.7261 - mean_squared_error: 0.0025 - val_loss: 0.
0019 - val_acc: 0.7150 - val_mean_squared_error: 0.0019
Epoch 42/50
Epoch 00041: val_loss improved from 0.00190 to 0.00186, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0024 - acc: 0.7313 - mean_squared_error: 0.0024 - val_loss: 0.
0019 - val_acc: 0.7126 - val_mean_squared_error: 0.0019
Epoch 43/50
Epoch 00042: val_loss did not improve
1s - loss: 0.0024 - acc: 0.7313 - mean_squared_error: 0.0024 - val_loss: 0.
0019 - val_acc: 0.7079 - val_mean_squared_error: 0.0019
Epoch 44/50
Epoch 00043: val_loss improved from 0.00186 to 0.00185, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0023 - acc: 0.7249 - mean_squared_error: 0.0023 - val_loss: 0.
0018 - val_acc: 0.7196 - val_mean_squared_error: 0.0018
Epoch 45/50
Epoch 00044: val_loss did not improve
1s - loss: 0.0023 - acc: 0.7319 - mean_squared_error: 0.0023 - val_loss: 0.
0019 - val_acc: 0.7220 - val_mean_squared_error: 0.0019
Epoch 46/50
Epoch 00045: val_loss improved from 0.00185 to 0.00181, saving model to ./m
```

```
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0022 - acc: 0.7249 - mean_squared_error: 0.0022 - val_loss: 0.
0018 - val_acc: 0.7290 - val_mean_squared_error: 0.0018
Epoch 47/50
Epoch 00046: val_loss improved from 0.00181 to 0.00171, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0022 - acc: 0.7278 - mean_squared_error: 0.0022 - val_loss: 0.
0017 - val_acc: 0.7220 - val_mean_squared_error: 0.0017
Epoch 48/50
Epoch 00047: val_loss improved from 0.00171 to 0.00169, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0021 - acc: 0.7354 - mean_squared_error: 0.0021 - val_loss: 0.
0017 - val_acc: 0.7266 - val_mean_squared_error: 0.0017
Epoch 49/50
Epoch 00048: val_loss did not improve
1s - loss: 0.0021 - acc: 0.7424 - mean_squared_error: 0.0021 - val_loss: 0.
0017 - val_acc: 0.7313 - val_mean_squared_error: 0.0017
Epoch 50/50
Epoch 00049: val_loss improved from 0.00169 to 0.00166, saving model to ./m
odel/Adam_model.weights.best.hdf5
1s - loss: 0.0021 - acc: 0.7296 - mean_squared_error: 0.0021 - val_loss: 0.
0017 - val_acc: 0.7407 - val_mean_squared_error: 0.0017



  Running model: dropout_base_model w/opt: Adamax
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.03776, saving model to ./mode
l/Adamax_model.weights.best.hdf5
3s - loss: 0.0297 - acc: 0.4188 - mean_squared_error: 0.0297 - val_loss: 0.
0378 - val_acc: 0.6963 - val_mean_squared_error: 0.0378
Epoch 2/50
Epoch 00001: val_loss improved from 0.03776 to 0.02951, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0114 - acc: 0.5362 - mean_squared_error: 0.0114 - val_loss: 0.
0295 - val_acc: 0.6963 - val_mean_squared_error: 0.0295
Epoch 3/50
Epoch 00002: val_loss improved from 0.02951 to 0.02418, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0089 - acc: 0.5970 - mean_squared_error: 0.0089 - val_loss: 0.
0242 - val_acc: 0.6963 - val_mean_squared_error: 0.0242
Epoch 4/50
Epoch 00003: val_loss improved from 0.02418 to 0.01630, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0080 - acc: 0.6092 - mean_squared_error: 0.0080 - val_loss: 0.
0163 - val_acc: 0.6963 - val_mean_squared_error: 0.0163
Epoch 5/50
Epoch 00004: val_loss did not improve
1s - loss: 0.0076 - acc: 0.6402 - mean_squared_error: 0.0076 - val_loss: 0.
0171 - val_acc: 0.6963 - val_mean_squared_error: 0.0171
Epoch 6/50
Epoch 00005: val_loss improved from 0.01630 to 0.01461, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0067 - acc: 0.6624 - mean_squared_error: 0.0067 - val_loss: 0.
0146 - val_acc: 0.6963 - val_mean_squared_error: 0.0146
Epoch 7/50
```

```
Epoch 00006: val_loss improved from 0.01461 to 0.01217, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0063 - acc: 0.6507 - mean_squared_error: 0.0063 - val_loss: 0.
0122 - val_acc: 0.6963 - val_mean_squared_error: 0.0122
Epoch 8/50
Epoch 00007: val_loss improved from 0.01217 to 0.01011, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0058 - acc: 0.6700 - mean_squared_error: 0.0058 - val_loss: 0.
0101 - val_acc: 0.7056 - val_mean_squared_error: 0.0101
Epoch 9/50
Epoch 00008: val_loss did not improve
1s - loss: 0.0055 - acc: 0.6700 - mean_squared_error: 0.0055 - val_loss: 0.
0115 - val_acc: 0.7009 - val_mean_squared_error: 0.0115
Epoch 10/50
Epoch 00009: val_loss did not improve
1s - loss: 0.0053 - acc: 0.6612 - mean_squared_error: 0.0053 - val_loss: 0.
0110 - val_acc: 0.7173 - val_mean_squared_error: 0.0110
Epoch 11/50
Epoch 00010: val_loss improved from 0.01011 to 0.00741, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0051 - acc: 0.6752 - mean_squared_error: 0.0051 - val_loss: 0.
0074 - val_acc: 0.7033 - val_mean_squared_error: 0.0074
Epoch 12/50
Epoch 00011: val_loss improved from 0.00741 to 0.00719, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0049 - acc: 0.6735 - mean_squared_error: 0.0049 - val_loss: 0.
0072 - val_acc: 0.7126 - val_mean_squared_error: 0.0072
Epoch 13/50
Epoch 00012: val_loss improved from 0.00719 to 0.00588, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0046 - acc: 0.6641 - mean_squared_error: 0.0046 - val_loss: 0.
0059 - val_acc: 0.7243 - val_mean_squared_error: 0.0059
Epoch 14/50
Epoch 00013: val_loss did not improve
1s - loss: 0.0045 - acc: 0.6834 - mean_squared_error: 0.0045 - val_loss: 0.
0096 - val_acc: 0.7173 - val_mean_squared_error: 0.0096
Epoch 15/50
Epoch 00014: val_loss improved from 0.00588 to 0.00504, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0044 - acc: 0.6904 - mean_squared_error: 0.0044 - val_loss: 0.
0050 - val_acc: 0.7079 - val_mean_squared_error: 0.0050
Epoch 16/50
Epoch 00015: val_loss did not improve
1s - loss: 0.0043 - acc: 0.6922 - mean_squared_error: 0.0043 - val_loss: 0.
0053 - val_acc: 0.7150 - val_mean_squared_error: 0.0053
Epoch 17/50
Epoch 00016: val_loss improved from 0.00504 to 0.00438, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0039 - acc: 0.6893 - mean_squared_error: 0.0039 - val_loss: 0.
0044 - val_acc: 0.7056 - val_mean_squared_error: 0.0044
Epoch 18/50
Epoch 00017: val_loss did not improve
1s - loss: 0.0041 - acc: 0.6822 - mean_squared_error: 0.0041 - val_loss: 0.
0056 - val_acc: 0.7079 - val_mean_squared_error: 0.0056
Epoch 19/50
Epoch 00018: val_loss improved from 0.00438 to 0.00361, saving model to ./m
odel/Adamax_model.weights.best.hdf5
```

```
1s - loss: 0.0038 - acc: 0.7091 - mean_squared_error: 0.0038 - val_loss: 0.
0036 - val_acc: 0.7126 - val_mean_squared_error: 0.0036
Epoch 20/50
Epoch 00019: val_loss did not improve
1s - loss: 0.0037 - acc: 0.6980 - mean_squared_error: 0.0037 - val_loss: 0.
0053 - val_acc: 0.7336 - val_mean_squared_error: 0.0053
Epoch 21/50
Epoch 00020: val_loss did not improve
1s - loss: 0.0037 - acc: 0.7091 - mean_squared_error: 0.0037 - val_loss: 0.
0063 - val_acc: 0.7290 - val_mean_squared_error: 0.0063
Epoch 22/50
Epoch 00021: val_loss improved from 0.00361 to 0.00325, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0036 - acc: 0.6910 - mean_squared_error: 0.0036 - val_loss: 0.
0032 - val_acc: 0.7266 - val_mean_squared_error: 0.0032
Epoch 23/50
Epoch 00022: val_loss improved from 0.00325 to 0.00308, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0034 - acc: 0.7091 - mean_squared_error: 0.0034 - val_loss: 0.
0031 - val_acc: 0.7126 - val_mean_squared_error: 0.0031
Epoch 24/50
Epoch 00023: val_loss improved from 0.00308 to 0.00279, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0034 - acc: 0.7056 - mean_squared_error: 0.0034 - val_loss: 0.
0028 - val_acc: 0.7383 - val_mean_squared_error: 0.0028
Epoch 25/50
Epoch 00024: val_loss did not improve
1s - loss: 0.0033 - acc: 0.7155 - mean_squared_error: 0.0033 - val_loss: 0.
0041 - val_acc: 0.7266 - val_mean_squared_error: 0.0041
Epoch 26/50
Epoch 00025: val_loss did not improve
1s - loss: 0.0032 - acc: 0.7208 - mean_squared_error: 0.0032 - val_loss: 0.
0032 - val_acc: 0.7150 - val_mean_squared_error: 0.0032
Epoch 27/50
Epoch 00026: val_loss did not improve
1s - loss: 0.0032 - acc: 0.7155 - mean_squared_error: 0.0032 - val_loss: 0.
0030 - val_acc: 0.7243 - val_mean_squared_error: 0.0030
Epoch 28/50
Epoch 00027: val_loss improved from 0.00279 to 0.00199, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0031 - acc: 0.7114 - mean_squared_error: 0.0031 - val_loss: 0.
0020 - val_acc: 0.7266 - val_mean_squared_error: 0.0020
Epoch 29/50
Epoch 00028: val_loss did not improve
1s - loss: 0.0031 - acc: 0.7266 - mean_squared_error: 0.0031 - val_loss: 0.
0026 - val_acc: 0.7266 - val_mean_squared_error: 0.0026
Epoch 30/50
Epoch 00029: val_loss did not improve
1s - loss: 0.0030 - acc: 0.7144 - mean_squared_error: 0.0030 - val_loss: 0.
0029 - val_acc: 0.7243 - val_mean_squared_error: 0.0029
Epoch 31/50
Epoch 00030: val_loss did not improve
1s - loss: 0.0029 - acc: 0.7383 - mean_squared_error: 0.0029 - val_loss: 0.
0032 - val_acc: 0.7243 - val_mean_squared_error: 0.0032
Epoch 32/50
Epoch 00031: val_loss did not improve
1s - loss: 0.0028 - acc: 0.7196 - mean_squared_error: 0.0028 - val_loss: 0.
```

```
0024 - val_acc: 0.7196 - val_mean_squared_error: 0.0024
Epoch 33/50
Epoch 00032: val_loss did not improve
1s - loss: 0.0027 - acc: 0.7103 - mean_squared_error: 0.0027 - val_loss: 0.
0029 - val_acc: 0.7336 - val_mean_squared_error: 0.0029
Epoch 34/50
Epoch 00033: val_loss did not improve
1s - loss: 0.0027 - acc: 0.7266 - mean_squared_error: 0.0027 - val_loss: 0.
0021 - val_acc: 0.7336 - val_mean_squared_error: 0.0021
Epoch 35/50
Epoch 00034: val_loss improved from 0.00199 to 0.00188, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0026 - acc: 0.7196 - mean_squared_error: 0.0026 - val_loss: 0.
0019 - val_acc: 0.7360 - val_mean_squared_error: 0.0019
Epoch 36/50
Epoch 00035: val_loss did not improve
1s - loss: 0.0025 - acc: 0.7366 - mean_squared_error: 0.0025 - val_loss: 0.
0026 - val_acc: 0.7336 - val_mean_squared_error: 0.0026
Epoch 37/50
Epoch 00036: val_loss improved from 0.00188 to 0.00160, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0025 - acc: 0.7319 - mean_squared_error: 0.0025 - val_loss: 0.
0016 - val_acc: 0.7336 - val_mean_squared_error: 0.0016
Epoch 38/50
Epoch 00037: val_loss did not improve
1s - loss: 0.0025 - acc: 0.7284 - mean_squared_error: 0.0025 - val_loss: 0.
0018 - val_acc: 0.7313 - val_mean_squared_error: 0.0018
Epoch 39/50
Epoch 00038: val_loss did not improve
1s - loss: 0.0024 - acc: 0.7296 - mean_squared_error: 0.0024 - val_loss: 0.
0022 - val_acc: 0.7313 - val_mean_squared_error: 0.0022
Epoch 40/50
Epoch 00039: val_loss did not improve
1s - loss: 0.0024 - acc: 0.7418 - mean_squared_error: 0.0024 - val_loss: 0.
0024 - val_acc: 0.7430 - val_mean_squared_error: 0.0024
Epoch 41/50
Epoch 00040: val_loss did not improve
1s - loss: 0.0023 - acc: 0.7430 - mean_squared_error: 0.0023 - val_loss: 0.
0018 - val_acc: 0.7313 - val_mean_squared_error: 0.0018
Epoch 42/50
Epoch 00041: val_loss did not improve
1s - loss: 0.0023 - acc: 0.7395 - mean_squared_error: 0.0023 - val_loss: 0.
0016 - val_acc: 0.7313 - val_mean_squared_error: 0.0016
Epoch 43/50
Epoch 00042: val_loss did not improve
1s - loss: 0.0022 - acc: 0.7442 - mean_squared_error: 0.0022 - val_loss: 0.
0019 - val_acc: 0.7313 - val_mean_squared_error: 0.0019
Epoch 44/50
Epoch 00043: val_loss did not improve
1s - loss: 0.0021 - acc: 0.7477 - mean_squared_error: 0.0021 - val_loss: 0.
0017 - val_acc: 0.7313 - val_mean_squared_error: 0.0017
Epoch 45/50
Epoch 00044: val_loss did not improve
1s - loss: 0.0021 - acc: 0.7407 - mean_squared_error: 0.0021 - val_loss: 0.
0017 - val_acc: 0.7336 - val_mean_squared_error: 0.0017
Epoch 46/50
Epoch 00045: val_loss did not improve
```

```
1s - loss: 0.0021 - acc: 0.7605 - mean_squared_error: 0.0021 - val_loss: 0.
0018 - val_acc: 0.7336 - val_mean_squared_error: 0.0018
Epoch 47/50
Epoch 00046: val_loss did not improve
1s - loss: 0.0020 - acc: 0.7395 - mean_squared_error: 0.0020 - val_loss: 0.
0019 - val_acc: 0.7430 - val_mean_squared_error: 0.0019
Epoch 48/50
Epoch 00047: val_loss improved from 0.00160 to 0.00156, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0020 - acc: 0.7576 - mean_squared_error: 0.0020 - val_loss: 0.
0016 - val_acc: 0.7383 - val_mean_squared_error: 0.0016
Epoch 49/50
Epoch 00048: val_loss did not improve
1s - loss: 0.0020 - acc: 0.7570 - mean_squared_error: 0.0020 - val_loss: 0.
0018 - val_acc: 0.7477 - val_mean_squared_error: 0.0018
Epoch 50/50
Epoch 00049: val_loss improved from 0.00156 to 0.00137, saving model to ./m
odel/Adamax_model.weights.best.hdf5
1s - loss: 0.0019 - acc: 0.7588 - mean_squared_error: 0.0019 - val_loss: 0.
0014 - val_acc: 0.7570 - val_mean_squared_error: 0.0014



  Running model: dropout_base_model w/opt: Nadam
Train on 1712 samples, validate on 428 samples
Epoch 1/50
Epoch 00000: val_loss improved from inf to 0.05018, saving model to ./mode
l/Nadam_model.weights.best.hdf5
3s - loss: 0.1321 - acc: 0.2728 - mean_squared_error: 0.1321 - val_loss: 0.
0502 - val_acc: 0.6963 - val_mean_squared_error: 0.0502
Epoch 2/50
Epoch 00001: val_loss improved from 0.05018 to 0.02929, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0192 - acc: 0.4731 - mean_squared_error: 0.0192 - val_loss: 0.
0293 - val_acc: 0.6963 - val_mean_squared_error: 0.0293
Epoch 3/50
Epoch 00002: val_loss did not improve
1s - loss: 0.0141 - acc: 0.5613 - mean_squared_error: 0.0141 - val_loss: 0.
0379 - val_acc: 0.6963 - val_mean_squared_error: 0.0379
Epoch 4/50
Epoch 00003: val_loss improved from 0.02929 to 0.01899, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0118 - acc: 0.6121 - mean_squared_error: 0.0118 - val_loss: 0.
0190 - val_acc: 0.6963 - val_mean_squared_error: 0.0190
Epoch 5/50
Epoch 00004: val_loss did not improve
1s - loss: 0.0112 - acc: 0.6332 - mean_squared_error: 0.0112 - val_loss: 0.
0206 - val_acc: 0.6963 - val_mean_squared_error: 0.0206
Epoch 6/50
Epoch 00005: val_loss improved from 0.01899 to 0.01631, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0098 - acc: 0.6361 - mean_squared_error: 0.0098 - val_loss: 0.
0163 - val_acc: 0.6963 - val_mean_squared_error: 0.0163
Epoch 7/50
Epoch 00006: val_loss did not improve
1s - loss: 0.0088 - acc: 0.6717 - mean_squared_error: 0.0088 - val_loss: 0.
0210 - val_acc: 0.6963 - val_mean_squared_error: 0.0210
```

```
Epoch 8/50
Epoch 00007: val_loss did not improve
1s - loss: 0.0083 - acc: 0.6618 - mean_squared_error: 0.0083 - val_loss: 0.
0170 - val_acc: 0.6963 - val_mean_squared_error: 0.0170
Epoch 9/50
Epoch 00008: val_loss improved from 0.01631 to 0.00938, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0076 - acc: 0.6595 - mean_squared_error: 0.0076 - val_loss: 0.
0094 - val_acc: 0.6963 - val_mean_squared_error: 0.0094
Epoch 10/50
Epoch 00009: val_loss did not improve
1s - loss: 0.0075 - acc: 0.6671 - mean_squared_error: 0.0075 - val_loss: 0.
0096 - val_acc: 0.6963 - val_mean_squared_error: 0.0096
Epoch 11/50
Epoch 00010: val_loss did not improve
1s - loss: 0.0070 - acc: 0.6741 - mean_squared_error: 0.0070 - val_loss: 0.
0116 - val_acc: 0.6963 - val_mean_squared_error: 0.0116
Epoch 12/50
Epoch 00011: val_loss did not improve
1s - loss: 0.0065 - acc: 0.6776 - mean_squared_error: 0.0065 - val_loss: 0.
0094 - val_acc: 0.6963 - val_mean_squared_error: 0.0094
Epoch 13/50
Epoch 00012: val_loss did not improve
1s - loss: 0.0064 - acc: 0.6875 - mean_squared_error: 0.0064 - val_loss: 0.
0100 - val_acc: 0.6963 - val_mean_squared_error: 0.0100
Epoch 14/50
Epoch 00013: val_loss improved from 0.00938 to 0.00739, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0060 - acc: 0.6922 - mean_squared_error: 0.0060 - val_loss: 0.
0074 - val_acc: 0.6963 - val_mean_squared_error: 0.0074
Epoch 15/50
Epoch 00014: val_loss did not improve
1s - loss: 0.0057 - acc: 0.6893 - mean_squared_error: 0.0057 - val_loss: 0.
0081 - val_acc: 0.6963 - val_mean_squared_error: 0.0081
Epoch 16/50
Epoch 00015: val_loss improved from 0.00739 to 0.00610, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0053 - acc: 0.6922 - mean_squared_error: 0.0053 - val_loss: 0.
0061 - val_acc: 0.6963 - val_mean_squared_error: 0.0061
Epoch 17/50
Epoch 00016: val_loss improved from 0.00610 to 0.00522, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0052 - acc: 0.6998 - mean_squared_error: 0.0052 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 18/50
Epoch 00017: val_loss did not improve
1s - loss: 0.0049 - acc: 0.6875 - mean_squared_error: 0.0049 - val_loss: 0.
0054 - val_acc: 0.6963 - val_mean_squared_error: 0.0054
Epoch 19/50
Epoch 00018: val_loss improved from 0.00522 to 0.00520, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0047 - acc: 0.6980 - mean_squared_error: 0.0047 - val_loss: 0.
0052 - val_acc: 0.6963 - val_mean_squared_error: 0.0052
Epoch 20/50
Epoch 00019: val_loss improved from 0.00520 to 0.00488, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0046 - acc: 0.6998 - mean_squared_error: 0.0046 - val_loss: 0.
```

```
0049 - val_acc: 0.6963 - val_mean_squared_error: 0.0049
Epoch 21/50
Epoch 00020: val_loss improved from 0.00488 to 0.00453, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0045 - acc: 0.6986 - mean_squared_error: 0.0045 - val_loss: 0.
0045 - val_acc: 0.6963 - val_mean_squared_error: 0.0045
Epoch 22/50
Epoch 00021: val_loss improved from 0.00453 to 0.00400, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0043 - acc: 0.7074 - mean_squared_error: 0.0043 - val_loss: 0.
0040 - val_acc: 0.6963 - val_mean_squared_error: 0.0040
Epoch 23/50
Epoch 00022: val_loss improved from 0.00400 to 0.00386, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0040 - acc: 0.7079 - mean_squared_error: 0.0040 - val_loss: 0.
0039 - val_acc: 0.7033 - val_mean_squared_error: 0.0039
Epoch 24/50
Epoch 00023: val_loss improved from 0.00386 to 0.00316, saving model to ./m
odel/Nadam_model.weights.best.hdf5
2s - loss: 0.0039 - acc: 0.7004 - mean_squared_error: 0.0039 - val_loss: 0.
0032 - val_acc: 0.7033 - val_mean_squared_error: 0.0032
Epoch 25/50
Epoch 00024: val_loss did not improve
1s - loss: 0.0037 - acc: 0.6992 - mean_squared_error: 0.0037 - val_loss: 0.
0032 - val_acc: 0.6963 - val_mean_squared_error: 0.0032
Epoch 26/50
Epoch 00025: val_loss improved from 0.00316 to 0.00304, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0036 - acc: 0.7132 - mean_squared_error: 0.0036 - val_loss: 0.
0030 - val_acc: 0.6986 - val_mean_squared_error: 0.0030
Epoch 27/50
Epoch 00026: val_loss improved from 0.00304 to 0.00292, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0034 - acc: 0.7068 - mean_squared_error: 0.0034 - val_loss: 0.
0029 - val_acc: 0.7056 - val_mean_squared_error: 0.0029
Epoch 28/50
Epoch 00027: val_loss did not improve
1s - loss: 0.0034 - acc: 0.7079 - mean_squared_error: 0.0034 - val_loss: 0.
0031 - val_acc: 0.7056 - val_mean_squared_error: 0.0031
Epoch 29/50
Epoch 00028: val_loss improved from 0.00292 to 0.00281, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0033 - acc: 0.7079 - mean_squared_error: 0.0033 - val_loss: 0.
0028 - val_acc: 0.7009 - val_mean_squared_error: 0.0028
Epoch 30/50
Epoch 00029: val_loss improved from 0.00281 to 0.00266, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0031 - acc: 0.7150 - mean_squared_error: 0.0031 - val_loss: 0.
0027 - val_acc: 0.7009 - val_mean_squared_error: 0.0027
Epoch 31/50
Epoch 00030: val_loss did not improve
1s - loss: 0.0031 - acc: 0.7074 - mean_squared_error: 0.0031 - val_loss: 0.
0027 - val_acc: 0.7079 - val_mean_squared_error: 0.0027
Epoch 32/50
Epoch 00031: val_loss improved from 0.00266 to 0.00263, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0030 - acc: 0.7097 - mean_squared_error: 0.0030 - val_loss: 0.
```

```
0026 - val_acc: 0.7009 - val_mean_squared_error: 0.0026
Epoch 33/50
Epoch 00032: val_loss improved from 0.00263 to 0.00260, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0030 - acc: 0.7202 - mean_squared_error: 0.0030 - val_loss: 0.
0026 - val_acc: 0.7009 - val_mean_squared_error: 0.0026
Epoch 34/50
Epoch 00033: val_loss improved from 0.00260 to 0.00244, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0029 - acc: 0.7120 - mean_squared_error: 0.0029 - val_loss: 0.
0024 - val_acc: 0.7056 - val_mean_squared_error: 0.0024
Epoch 35/50
Epoch 00034: val_loss did not improve
1s - loss: 0.0029 - acc: 0.7114 - mean_squared_error: 0.0029 - val_loss: 0.
0025 - val_acc: 0.7056 - val_mean_squared_error: 0.0025
Epoch 36/50
Epoch 00035: val_loss improved from 0.00244 to 0.00231, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0028 - acc: 0.7114 - mean_squared_error: 0.0028 - val_loss: 0.
0023 - val_acc: 0.7033 - val_mean_squared_error: 0.0023
Epoch 37/50
Epoch 00036: val_loss did not improve
1s - loss: 0.0027 - acc: 0.7120 - mean_squared_error: 0.0027 - val_loss: 0.
0024 - val_acc: 0.7009 - val_mean_squared_error: 0.0024
Epoch 38/50
Epoch 00037: val_loss improved from 0.00231 to 0.00228, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0027 - acc: 0.7138 - mean_squared_error: 0.0027 - val_loss: 0.
0023 - val_acc: 0.7079 - val_mean_squared_error: 0.0023
Epoch 39/50
Epoch 00038: val_loss did not improve
1s - loss: 0.0027 - acc: 0.7155 - mean_squared_error: 0.0027 - val_loss: 0.
0024 - val_acc: 0.7056 - val_mean_squared_error: 0.0024
Epoch 40/50
Epoch 00039: val_loss did not improve
1s - loss: 0.0026 - acc: 0.7179 - mean_squared_error: 0.0026 - val_loss: 0.
0023 - val_acc: 0.7056 - val_mean_squared_error: 0.0023
Epoch 41/50
Epoch 00040: val_loss improved from 0.00228 to 0.00220, saving model to ./m
odel/Nadam_model.weights.best.hdf5
2s - loss: 0.0026 - acc: 0.7173 - mean_squared_error: 0.0026 - val_loss: 0.
0022 - val_acc: 0.7079 - val_mean_squared_error: 0.0022
Epoch 42/50
Epoch 00041: val_loss improved from 0.00220 to 0.00219, saving model to ./m
odel/Nadam_model.weights.best.hdf5
2s - loss: 0.0026 - acc: 0.7185 - mean_squared_error: 0.0026 - val_loss: 0.
0022 - val_acc: 0.7056 - val_mean_squared_error: 0.0022
Epoch 43/50
Epoch 00042: val_loss improved from 0.00219 to 0.00215, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0025 - acc: 0.7155 - mean_squared_error: 0.0025 - val_loss: 0.
0021 - val_acc: 0.7056 - val_mean_squared_error: 0.0021
Epoch 44/50
Epoch 00043: val_loss improved from 0.00215 to 0.00210, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0025 - acc: 0.7196 - mean_squared_error: 0.0025 - val_loss: 0.
0021 - val_acc: 0.7079 - val_mean_squared_error: 0.0021
```

```
Epoch 45/50
Epoch 00044: val_loss did not improve
1s - loss: 0.0025 - acc: 0.7109 - mean_squared_error: 0.0025 - val_loss: 0.
0022 - val_acc: 0.7033 - val_mean_squared_error: 0.0022
Epoch 46/50
Epoch 00045: val_loss improved from 0.00210 to 0.00202, saving model to ./m
odel/Nadam_model.weights.best.hdf5
2s - loss: 0.0025 - acc: 0.7266 - mean_squared_error: 0.0025 - val_loss: 0.
0020 - val_acc: 0.7079 - val_mean_squared_error: 0.0020
Epoch 47/50
Epoch 00046: val_loss did not improve
1s - loss: 0.0024 - acc: 0.7120 - mean_squared_error: 0.0024 - val_loss: 0.
0021 - val_acc: 0.7033 - val_mean_squared_error: 0.0021
Epoch 48/50
Epoch 00047: val_loss improved from 0.00202 to 0.00201, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0024 - acc: 0.7109 - mean_squared_error: 0.0024 - val_loss: 0.
0020 - val_acc: 0.7079 - val_mean_squared_error: 0.0020
Epoch 49/50
Epoch 00048: val_loss improved from 0.00201 to 0.00195, saving model to ./m
odel/Nadam_model.weights.best.hdf5
2s - loss: 0.0024 - acc: 0.7167 - mean_squared_error: 0.0024 - val_loss: 0.
0020 - val_acc: 0.7033 - val_mean_squared_error: 0.0020
Epoch 50/50
Epoch 00049: val_loss improved from 0.00195 to 0.00193, saving model to ./m
odel/Nadam_model.weights.best.hdf5
1s - loss: 0.0024 - acc: 0.7167 - mean_squared_error: 0.0024 - val_loss: 0.
0019 - val_acc: 0.7056 - val_mean_squared_error: 0.0019
```

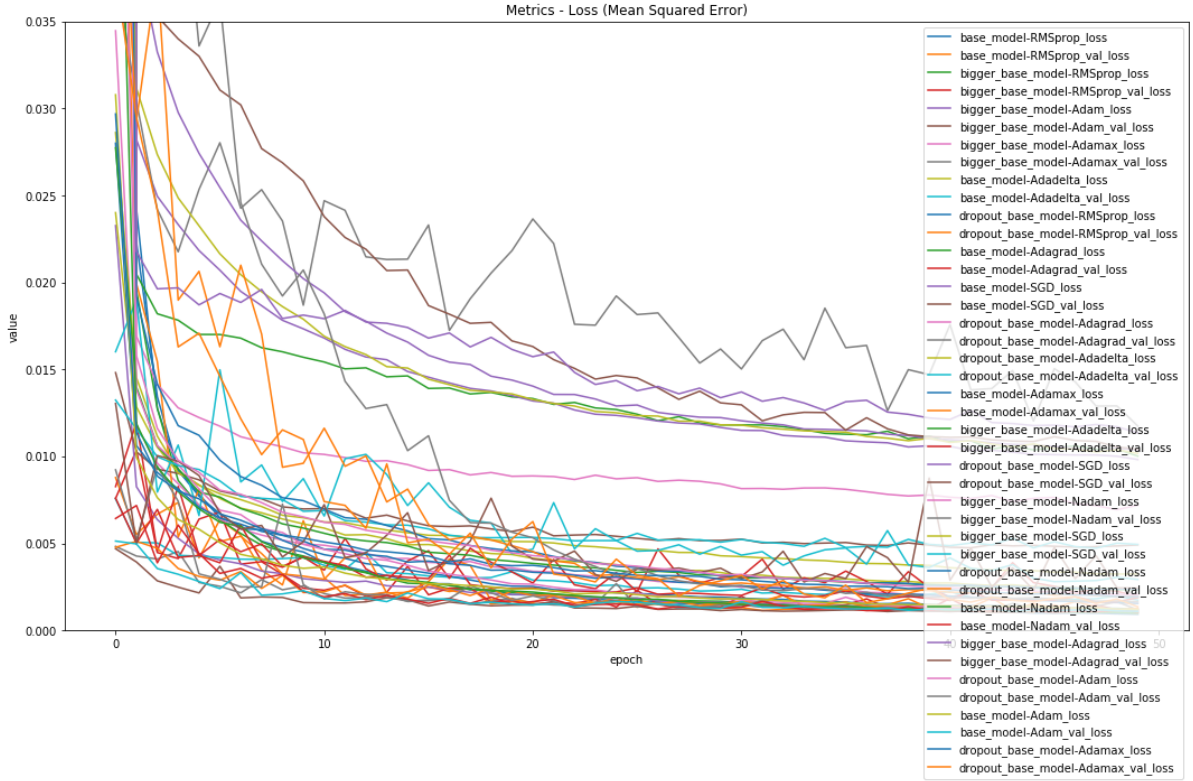
Note: If rerun, the section below should use the validation set loss instead of the training set loss to choose the best performing model. As it is, the current model choosen performs well for the task and the full model history was not preserved. Re-fitting the model is a time-expensive operation, and the added value does not currently justify it.

In [29]:
```
best_MSE=float("inf")
print(trained_models.keys())
for m_name in model_names:
    for opt in optimizers:
        curr_model_name=m_name+'-'+opt
        hist,model=trained_models[curr_model_name]
        #print(hist.history.keys())
        print(curr_model_name,hist.history['loss'][-1])
        if min(hist.history['loss']) <best_MSE:
            best_MSE=hist.history['loss'][-1]
            best_model_opt=curr_model_name
print('\nBest Opt:',best_model_opt,'w/MSE of:',best_MSE)

#set the best model as the one we use
hist,model=trained_models[best_model_opt]

## TODO: Save the model as model.h5
# Save history into a pickle file.
import pickle
try:
    del hist.model #needed to preserve history via pickle
except:
    print('no prob') # run things out of order and there is no hist.model to d
elete, that's not a problem
with open(model_path+"hist.pkl", 'wb') as hist_file_handle:
    pickle.dump(hist, hist_file_handle, protocol=pickle.HIGHEST_PROTOCOL)

model.save(model_path+'best_model.h5')
```

```
dict_keys(['base_model-RMSprop', 'bigger_base_model-RMSprop', 'bigger_base_mo
del-Adam', 'bigger_base_model-Adamax', 'base_model-Adadelta', 'dropout_base_m
odel-RMSprop', 'base_model-Adagrad', 'base_model-SGD', 'dropout_base_model-Ad
agrad', 'dropout_base_model-Adadelta', 'base_model-Adamax', 'bigger_base_mode
l-Adadelta', 'dropout_base_model-SGD', 'bigger_base_model-Nadam', 'bigger_bas
e_model-SGD', 'dropout_base_model-Nadam', 'base_model-Nadam', 'bigger_base_mo
del-Adagrad', 'dropout_base_model-Adam', 'base_model-Adam', 'dropout_base_mod
el-Adamax'])
base_model-SGD 0.00982422917803
base_model-RMSprop 0.000932490296453
base_model-Adagrad 0.0100213121536
base_model-Adadelta 0.00237443998117
base_model-Adam 0.00120959542888
base_model-Adamax 0.00155757706956
base_model-Nadam 0.000997994311598
bigger_base_model-SGD 0.0101803487393
bigger_base_model-RMSprop 0.00103292662855
bigger_base_model-Adagrad 0.0113620946236
bigger_base_model-Adadelta 0.00234861237815
bigger_base_model-Adam 0.00107691402299
bigger_base_model-Adamax 0.00155688610546
bigger_base_model-Nadam 0.00114658472746
dropout_base_model-SGD 0.0103783547513
dropout_base_model-RMSprop 0.00123239942896
dropout_base_model-Adagrad 0.00719947848365
dropout_base_model-Adadelta 0.00324772258951
dropout_base_model-Adam 0.00205891006017
dropout_base_model-Adamax 0.00194159019782
dropout_base_model-Nadam 0.00237174390458

Best Opt: base_model-RMSprop w/MSE of: 0.000932490296453
```

In [45]:
```
## TODO: Visualize the training and validation loss of your neural network

legend = []
plt.figure(figsize=(18, 10))

model_keys=trained_models.keys()
for model_key in model_keys:
    opt_hist,opt_model=trained_models[model_key]
    for key, metric in opt_hist.history.items():
        if 'acc' in key and 'val' in key:
            plt.plot(metric)
            legend.append(model_key+'_'+key)

plt.title('Metrics - Accuracy, All models')
plt.ylabel('value')
plt.xlabel('epoch')
plt.legend(legend, loc='lower right')
plt.ylim(0.68, 0.85)
plt.show()
```

In [35]:
```
## TODO: Visualize the training and validation loss of your neural network

legend = []
plt.figure(figsize=(18, 10))


model_keys=trained_models.keys()
for model_key in model_keys:
    opt_hist,opt_model=trained_models[model_key]
    for key, metric in opt_hist.history.items():
        if 'loss' in key:
            plt.plot(metric)
            legend.append(model_key+'_'+key)

plt.title('Metrics - Loss (Mean Squared Error), All models')
plt.ylabel('value')
plt.xlabel('epoch')
plt.legend(legend, loc='upper right')
plt.ylim(0, 0.035)
plt.show()
```



Step 7: Visualize the Loss and Test Predictions

(IMPLEMENTATION) Answer a few questions and visualize the loss

Question 1: Outline the steps you took to get to your final neural network architecture and your reasoning at each step.

Answer: My base model is extended from the Convolutional Neural Net(CNN) model I developed during the CNN and dog breed projects, adapted to our conditions. This fairly basic CNN model is both fast performing and fast training, and able to acheive reasonable results. I also extended this model with a version with more dropout, and a model with a bigger kernel. The final model was selected based on model performance. This was facilitated by running each model, via the Amazon cloud, on a machine with GPU computation resources. This allowed each model trial of 50 epochs to be to run in approximatly 5 mins-10, as opposed to over an hour on CPU. The final model was the initial configuration I used, with some minor changes to the values for dropout.

Question 2: Defend your choice of optimizer. Which optimizers did you test, and how did you determine which worked best?

Answer: Optimizer selection was done by choosing the winner from the a train run on each, and selecting the model that had best loss performance, in this case using RMS. It can be observed that it performed better in every version of the model, with only SGD performing at a similar level. More opportunity exists for selection of the optimizer as, each was used with default inputs, and hyperparameter optimization may have lead to different optimizer selection, and better overall results. In this case, the model has performed well enough that further optimization wasn't needed to see good results.

Use the code cell below to plot the training and validation loss of your neural network. You may find this resource (http://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/) useful.

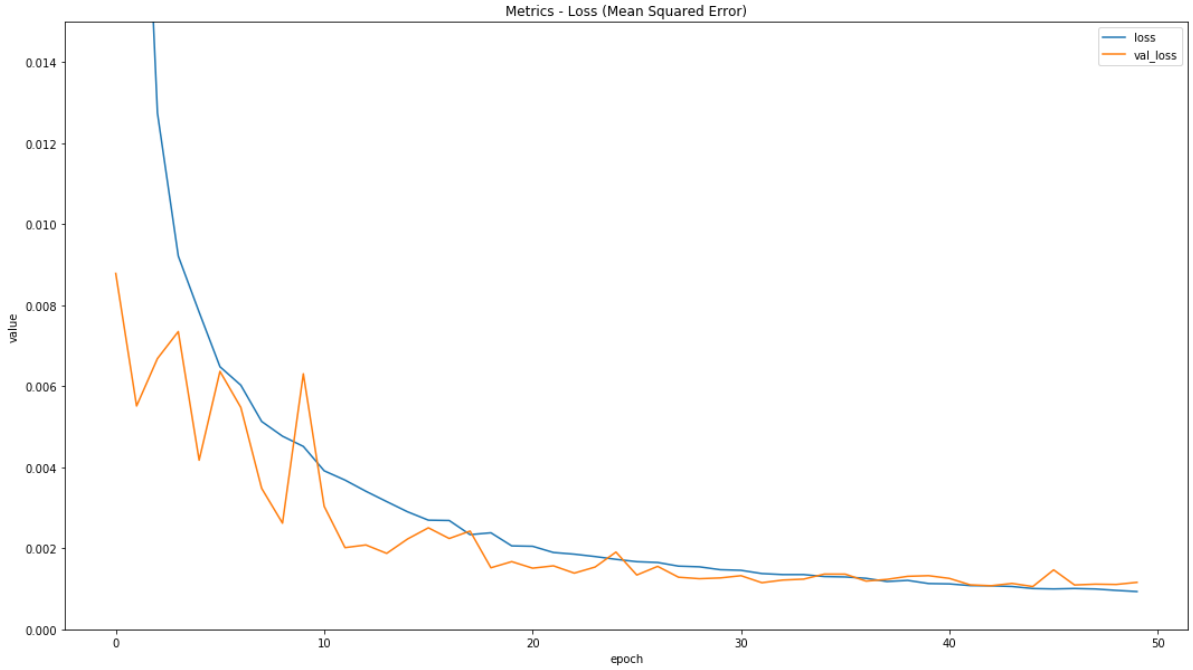
Question 3: Do you notice any evidence of overfitting or underfitting in the above plot? If so, what steps have you taken to improve your model? Note that slight overfitting or underfitting will not hurt your chances of a successful submission, as long as you have attempted some solutions towards improving your model (such as *regularization, dropout, increased/decreased number of layers, etc*).

Answer:

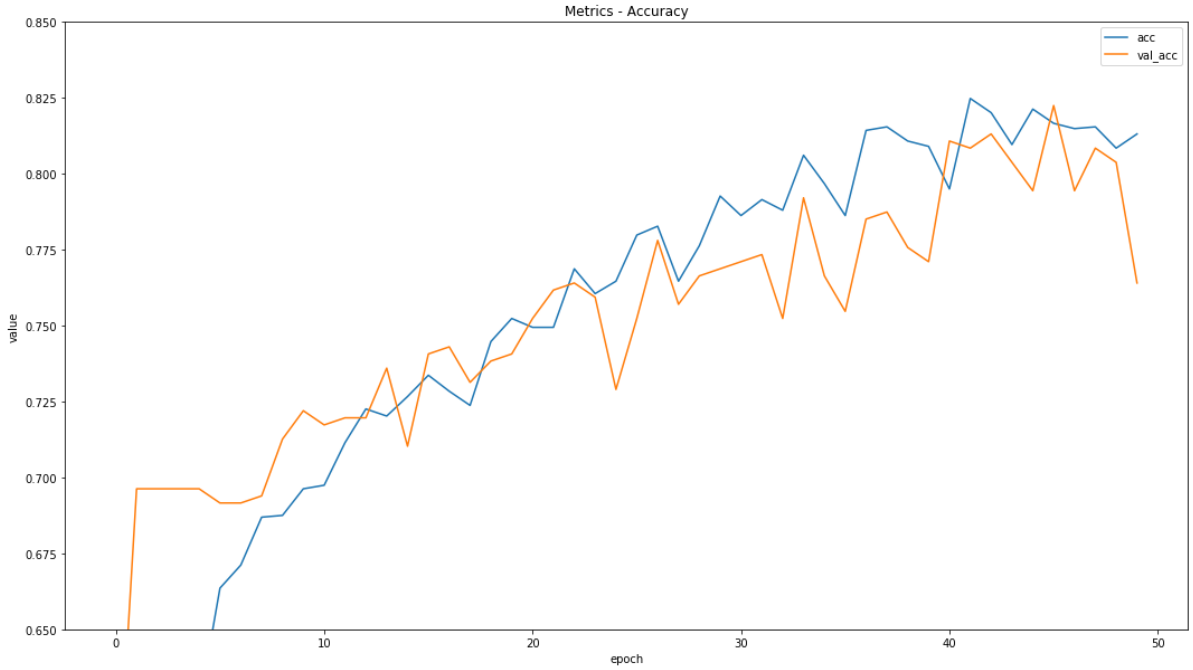
I have taken some measures to assist in preventing overfitting (specifically, we have added dropout). As training progresses, we do see some small indication of overfiting, specifically the loss continuing to decrease as validation increases, but the effect is currently minimal. More experimentation would be needed to determine if this trend would continue in the choosen model. If found to be true, the first step may be to augment the training data, with various blurred/sharpened, and scewed versions of the current trainging data set. It may also be possible to add back in the unused portion of our data that was missing features

In [ ]:
```
#reload the model/hist from disk, so we don't have to rerun
import pickle
with open(model_path+"hist.pkl", 'rb') as handle:
    hist = pickle.load(handle)

model=load_model(model_path+'best_model.h5') # issues seen with this, but seem
s to load the model despite errors.
```

In [391]:
```
legend = []
plt.figure(figsize=(18, 10))

for key, metric in hist.history.items():
    if 'loss' in key:
        plt.plot(metric)
        legend.append(key)

plt.title('Metrics - Loss (Mean Squared Error)')
plt.ylabel('value')
plt.xlabel('epoch')
plt.legend(legend, loc='upper right')
plt.ylim(0, 0.015)
plt.show()
```



In [390]:
```
legend = []
plt.figure(figsize=(18, 10))

for key, metric in hist.history.items():
    if 'acc' in key:
        plt.plot(metric)
        legend.append(key)

plt.title('Metrics - Accuracy')
plt.ylabel('value')
plt.xlabel('epoch')
plt.legend(legend, loc='upper right')
plt.ylim(0.65, 0.85)
plt.show()
```



Visualize a Subset of the Test Predictions

Execute the code cell below to visualize your model's predicted keypoints on a subset of the testing images.

In [36]:
```
y_test = model.predict(X_test)
fig = plt.figure(figsize=(20,20))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05
)
for i in range(9):
    ax = fig.add_subplot(3, 3, i + 1, xticks=[], yticks=[])
    plot_data(X_test[i], y_test[i], ax)
```

Step 8: Complete the pipeline

With the work you did in Sections 1 and 2 of this notebook, along with your freshly trained facial keypoint detector, you can now complete the full pipeline. That is given a color image containing a person or persons you can now

- Detect the faces in this image automatically using OpenCV
- Predict the facial keypoints in each face detected in the image
- Paint predicted keypoints on each face detected

In this Subsection you will do just this!

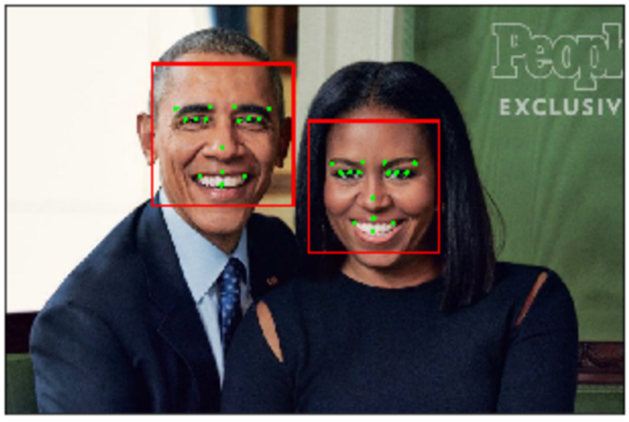
(IMPLEMENTATION) Facial Keypoints Detector

Use the OpenCV face detection functionality you built in previous Sections to expand the functionality of your keypoints detector to color images with arbitrary size. Your function should perform the following steps

1. Accept a color image.
2. Convert the image to grayscale.
3. Detect and crop the face contained in the image.
4. Locate the facial keypoints in the cropped image.
5. Overlay the facial keypoints in the original (color, uncropped) image.

Note: step 4 can be the trickiest because remember your convolutional network is only trained to detect facial keypoints in 96×96 grayscale images where each pixel was normalized to lie in the interval $[0, 1]$, and remember that each facial keypoint was normalized during training to the interval $[-1, 1]$. This means - practically speaking - to paint detected keypoints onto a test face you need to perform this same pre-processing to your candidate face - that is after detecting it you should resize it to 96×96 and normalize its values before feeding it into your facial keypoint detector. To be shown correctly on the original image the output keypoints from your detector then need to be shifted and re-normalized from the interval $[-1, 1]$ to the width and height of your detected face.

When complete you should be able to produce example images like the one below



In [258]:
```
# Load in color image for face detection
image = cv2.imread('images/obamas4.jpg')


# Convert the image to RGB colorspace
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

image_copy = np.copy(image)

# plot our image
fig = plt.figure(figsize = (9,9))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.set_title('image copy')
ax1.imshow(image_copy)
```

Out[258]: <matplotlib.image.AxesImage at 0x205f78a6400>

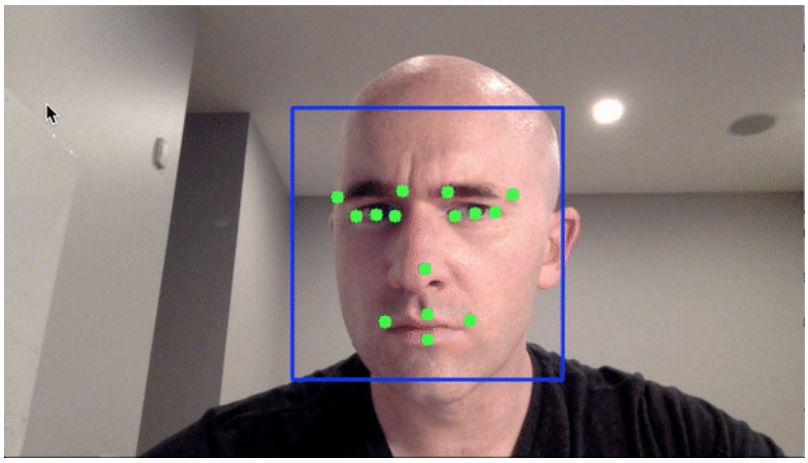
In [262]:
```
### TODO: Use the face detection code we saw in Section 1 with your trained co
nv-net
## TODO : Paint the predicted keypoints on the test image


## TODO: Implement face detection

## TODO: Blur the bounding box around each detected face using an averaging fi
lter and display the result
def detect_faces_add_keypoints(frame):
    frame_copy=np.copy(frame)
    gray_frame = cv2.cvtColor(frame_copy, cv2.COLOR_RGB2GRAY)
    other_detector='norm'
    #following works for my webcams
    face_locations = face_cascade.detectMultiScale(gray_frame, 1.1, 10)
    if len(face_locations)==0: # we didn't detect anything, let's try with som
e different settings
        # following works well for the Obama pics
        other_detector='alt'
        face_locations = face_cascade.detectMultiScale(gray_frame, 1.25, 6)

    if len(face_locations)>0:
        # for each detected face, resize and match the model input.
        faces_prep=[]
        for (x, y, w, h) in face_locations:
            #cv2.rectangle(frame_copy, (x, y), (x+w, y+h), (0, 255, 0), 2) # t
o highlight the face areas for debugging
            gray_face = gray_frame[y:y+h,x:x+w] #portion of image containing f
ace
            gray_resized_face = cv2.resize(gray_face, (96, 96)) # resize to ma
tch model train images
            gray_resized_face_dim_added = np.expand_dims(gray_resized_face, ax
is=2) # ditto
            faces_prep.append(gray_resized_face_dim_added / 255) #normalize pi
xel values

        # for all faces found, detect key points
        faces_key_points = model.predict(np.asarray(faces_prep))

        #for the detected faces/keypoints, map kp to original image
        for i in range(len(face_locations)):
            x,y,w,h=face_locations[i]
            face_x_KP=(faces_key_points[i][0::2]*48+48)*w/96+x
            face_y_KP=(faces_key_points[i][1::2]*48+48)*h/96+y
            for i in range(len(face_x_KP)):
                x_kp=face_x_KP[i]
                y_kp=face_y_KP[i]
                cv2.circle(frame_copy, (x_kp, y_kp), 3, (0, 255, 0), -1)
    #put debug text in pic, as needed
    font = cv2.FONT_HERSHEY_SIMPLEX
    display_text='Num_faces:'+str(len(face_locations))+' Detector_used:'+other
_detector
    cv2.putText(frame_copy,display_text,(10,30), font, 1,(20,20,255),2,cv2.LIN
E_AA)
    return frame_copy
```

In [269]:
```
#run face detect on saved image
image_kp_face=detect_faces_add_keypoints(image)

# Display the image
fig = plt.figure(figsize = (10,10))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.set_title('Facial Key_Point Image')
ax1.imshow(image_kp_face)
```

Out[269]: <matplotlib.image.AxesImage at 0x205bca58f28>

(Optional) Further Directions - add a filter using facial keypoints to your laptop camera

Now you can add facial keypoint detection to your laptop camera - as illustrated in the gif below.



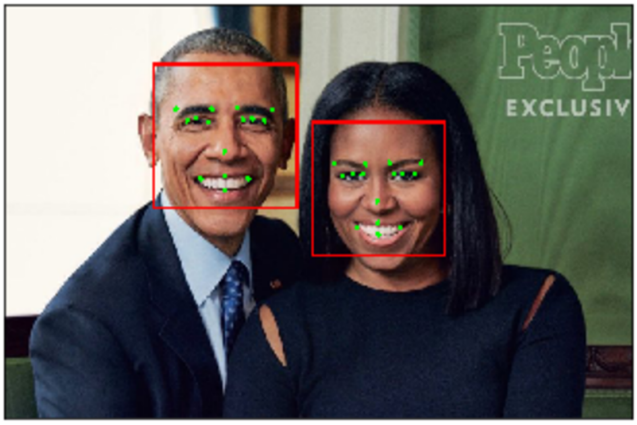
The next Python cell contains the basic laptop video camera function used in the previous optional video exercises. Combine it with the functionality you developed for keypoint detection and marking in the previous exercise and you should be good to go!

In [272]:
```
import cv2
import time
from keras.models import load_model
def laptop_camera_go():
    # Create instance of video capturer
    cv2.namedWindow("face detection activated")
    # I am on a laptop and have two cams....why not use the best availalbe?
    try:
        vc = cv2.VideoCapture(0)
    except:
        vc = cv2.VideoCapture(0)

    # Try to get the first frame
    if vc.isOpened():
        rval, frame = vc.read()
    else:
        rval = False

    # keep video stream open
    while rval:
        # Convert the image to RGB colorspace
        frame = detect_faces_add_keypoints(frame)

        # plot image from camera with detections marked
        cv2.imshow("face detection activated", detect_faces_add_keypoints(fram
e))

        # exit functionality - press any key to exit laptop video
        key = cv2.waitKey(20)
        if key > 0: # exit by pressing any key
            # destroy windows
            cv2.destroyAllWindows()

            # hack from stack overflow for making sure window closes on osx --
> https://stackoverflow.com/questions/6116564/destroywindow-does-not-close-win
dow-on-mac-using-python-and-opencv
            for i in range (1,5):
                cv2.waitKey(1)
            return

        # read next frame
        time.sleep(0.05)             # control framerate for computation - def
ault 20 frames per sec
        rval, frame = vc.read()
```

In [273]:
```
# Run your keypoint face painter
# The face cascade was returning old info frm the obama's pic instead of the w
ebcam pic, at random intervals
# This seems to reset it such that it works
face_cascade = cv2.CascadeClassifier('detector_architectures/haarcascade_front
alface_default.xml')

laptop_camera_go()
```

(Optional) Further Directions - add a filter using facial keypoints

Using your freshly minted facial keypoint detector pipeline you can now do things like add fun filters to a person's face automatically. In this optional exercise you can play around with adding sunglasses automatically to each individual's face in an image as shown in a demonstration image below.



To produce this effect an image of a pair of sunglasses shown in the Python cell below.

In [275]:
```
# Load in sunglasses image - note the usage of the special option
# cv2.IMREAD_UNCHANGED, this option is used because the sunglasses
# image has a 4th channel that allows us to control how transparent each pixel
 in the image is
sunglasses = cv2.imread("images/sunglasses_4.png", cv2.IMREAD_UNCHANGED)

# Plot the image
fig = plt.figure(figsize = (6,6))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.imshow(sunglasses)
ax1.axis('off');
```

This image is placed over each individual's face using the detected eye points to determine the location of the sunglasses, and eyebrow points to determine the size that the sunglasses should be for each person (one could also use the nose point to determine this).

Notice that this image actually has *4 channels*, not just 3.

In [276]:
```
# Print out the shape of the sunglasses image
print ('The sunglasses image has shape: ' + str(np.shape(sunglasses)))
```

```
The sunglasses image has shape: (1123, 3064, 4)
```

It has the usual red, blue, and green channels any color image has, with the 4th channel representing the transparency level of each pixel in the image. Here's how the transparency channel works: the lower the value, the more transparent the pixel will become. The lower bound (completely transparent) is zero here, so any pixels set to 0 will not be seen.

This is how we can place this image of sunglasses on someone's face and still see the area around of their face where the sunglasses lie - because these pixels in the sunglasses image have been made completely transparent.

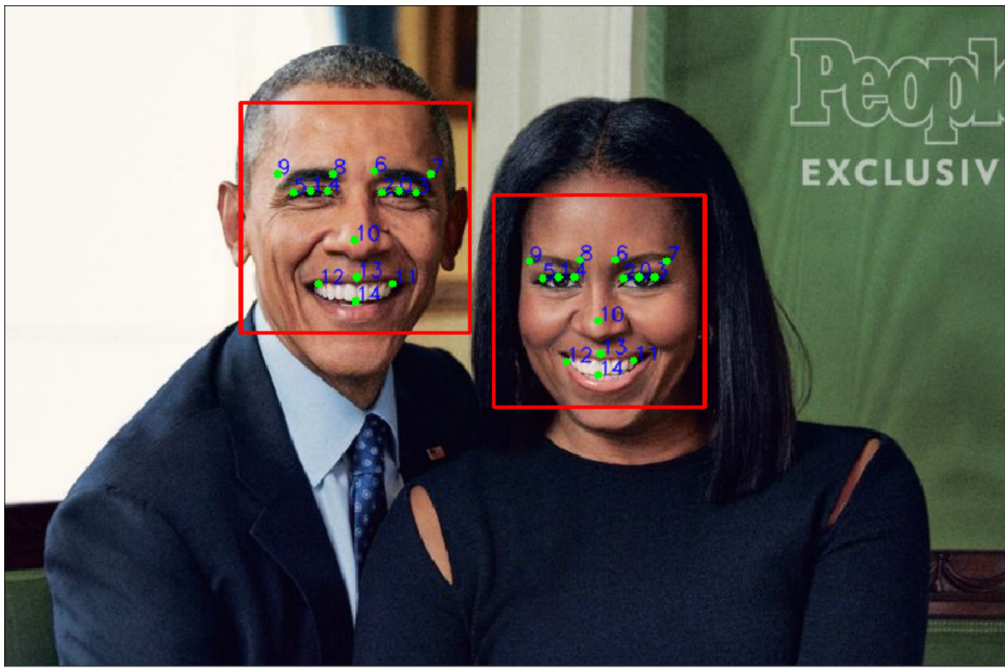
Lets check out the alpha channel of our sunglasses image in the next Python cell. Note because many of the pixels near the boundary are transparent we'll need to explicitly print out non-zero values if we want to see them.

In [277]:
```
# Print out the sunglasses transparency (alpha) channel
alpha_channel = sunglasses[:,:,3]
print ('the alpha channel here looks like')
print (alpha_channel)

# Just to double check that there are indeed non-zero values
# Let's find and print out every value greater than zero
values = np.where(alpha_channel != 0)
print ('\n the non-zero values of the alpha channel look like')
print (values)
```

```
the alpha channel here looks like
[[0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]
 ...,
 [0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]]

 the non-zero values of the alpha channel look like
(array([  17,   17,   17, ..., 1109, 1109, 1109], dtype=int64), array([ 687,
688,  689, ..., 2376, 2377, 2378], dtype=int64))
```

This means that when we place this sunglasses image on top of another image, we can use the transparency channel as a filter to tell us which pixels to overlay on a new image (only the non-transparent ones with values greater than zero).

One last thing: it's helpful to understand which keypoint belongs to the eyes, mouth, etc. So, in the image below, we also display the index of each facial keypoint directly on the image so that you can tell which keypoints are for the eyes, eyebrows, etc.



With this information, you're well on your way to completing this filtering task! See if you can place the sunglasses automatically on the individuals in the image loaded in / shown in the next Python cell.

In [278]:
```
# Load in color image for face detection
image = cv2.imread('images/obamas4.jpg')

# Convert the image to RGB colorspace
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)


# Plot the image
fig = plt.figure(figsize = (8,8))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.set_title('Original Image')
ax1.imshow(image)
```

Out[278]: <matplotlib.image.AxesImage at 0x205bbb65470>

In [299]:
```
# function to overlay a transparent image on background.
# https://pytech-solution.blogspot.com/2017/07/alphablending.html
def transparentOverlay(src , overlay , pos=(0,0),scale = 1):
    """
    :param src: Input Color Background Image
    :param overlay: transparent Image (BGRA)
    :param pos:  position where the image to be blit.
    :param scale : scale factor of transparent image.
    :return: Resultant Image
    """
    overlay = cv2.resize(overlay,(0,0),fx=scale,fy=scale)
    h,w,_ = overlay.shape  # Size of foreground
    rows,cols,_ = src.shape  # Size of background Image
    y,x = pos[0],pos[1]    # Position of foreground/overlay image

    #loop over all pixels and apply the blending equation
    for i in range(h):
        for j in range(w):
            if x+i >= rows or y+j >= cols:
                continue
            alpha = float(overlay[i][j][3]/255) # read the alpha channel
            src[x+i][y+j] = alpha*overlay[i][j][:3]+(1-alpha)*src[x+i][y+j]
    return src
```

In [369]:
```
## (Optional) TODO: Use the face detection code we saw in Section 1 with you
r trained conv-net to put
## sunglasses on the individuals in our test image

def detect_faces_add_glasses(frame):
    frame_copy=np.copy(frame)
    gray_frame = cv2.cvtColor(frame_copy, cv2.COLOR_RGB2GRAY)
    other_detector='norm'
    #following works for my webcams
    face_locations = face_cascade.detectMultiScale(gray_frame, 1.1, 10)
    if len(face_locations)==0: # we didn't detect anything, let's try with s
ome different settings
        # following works well for the Obama pics
        other_detector='alt'
        face_locations = face_cascade.detectMultiScale(gray_frame, 1.25, 6)


    if len(face_locations)>0:
        # for each detected face, resize and match the model input.
        faces_prep=[]
        for (x, y, w, h) in face_locations:
            #cv2.rectangle(frame_copy, (x, y), (x+w, y+h), (0, 255, 0), 2) #
 to highlight the face areas for debugging
            gray_face = gray_frame[y:y+h,x:x+w] #portion of image containing
 face
            gray_resized_face = cv2.resize(gray_face, (96, 96)) # resize to
 match model train images
            gray_resized_face_dim_added = np.expand_dims(gray_resized_face,
axis=2) # ditto
            faces_prep.append(gray_resized_face_dim_added / 255) #normalize
 pixel values
```

```
        # for all faces found, detect key points
        faces_key_points = model.predict(np.asarray(faces_prep))

        #for the detected faces/keypoints, map kp to original image
        for i in range(len(face_locations)):
            x,y,w,h=face_locations[i]
            face_x_KP=(faces_key_points[i][0::2]*48+48)*w/96+x
            face_y_KP=(faces_key_points[i][1::2]*48+48)*h/96+y

            glasses_width=face_x_KP[7]-face_x_KP[9] #dist between outer eyeb
row

            glasses_height=face_y_KP[10]-face_y_KP[9] #dist outer eyebrow to
 nose
            scale=(glasses_width/np.shape(sunglasses)[1])*1.2
            glasses_x_diff=((glasses_width*scale))

            frame_copy=transparentOverlay(frame_copy,sunglasses,(face_x_KP[9
]-glasses_x_diff,face_y_KP[8] ),scale)
            # for debugging, show the keypoints
            for i in range(len(face_x_KP)):
                x_kp=face_x_KP[i]
                y_kp=face_y_KP[i]
                #cv2.circle(frame_copy, (x_kp, y_kp), 3, (0, 255, 0), -1)
    #put debug text in pic, as needed
    font = cv2.FONT_HERSHEY_SIMPLEX
    display_text='Num_faces:'+str(len(face_locations))+' Detector_used:'+oth
er_detector
    cv2.putText(frame_copy,display_text,(10,30), font, 1,(20,20,255),2,cv2.L
INE_AA)
    return frame_copy
```

In [370]:
```
#run face detect on saved image
image_kp_face=detect_faces_add_glasses(image)

# Display the image
fig = plt.figure(figsize = (10,10))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.set_title('Facial Key_Point Image')
ax1.imshow(image_kp_face)
```

```
C:\Users\llathrop\AppData\Local\conda\conda\envs\aind-dog\lib\site-packages\i
pykernel\__main__.py:21: VisibleDeprecationWarning: using a non-integer numbe
r instead of an integer will result in an error in the future
```

Out[370]:
```
<matplotlib.image.AxesImage at 0x205c182e710>
```

(Optional) Further Directions - add a filter using facial keypoints to your laptop camera

Now you can add the sunglasses filter to your laptop camera - as illustrated in the gif below.

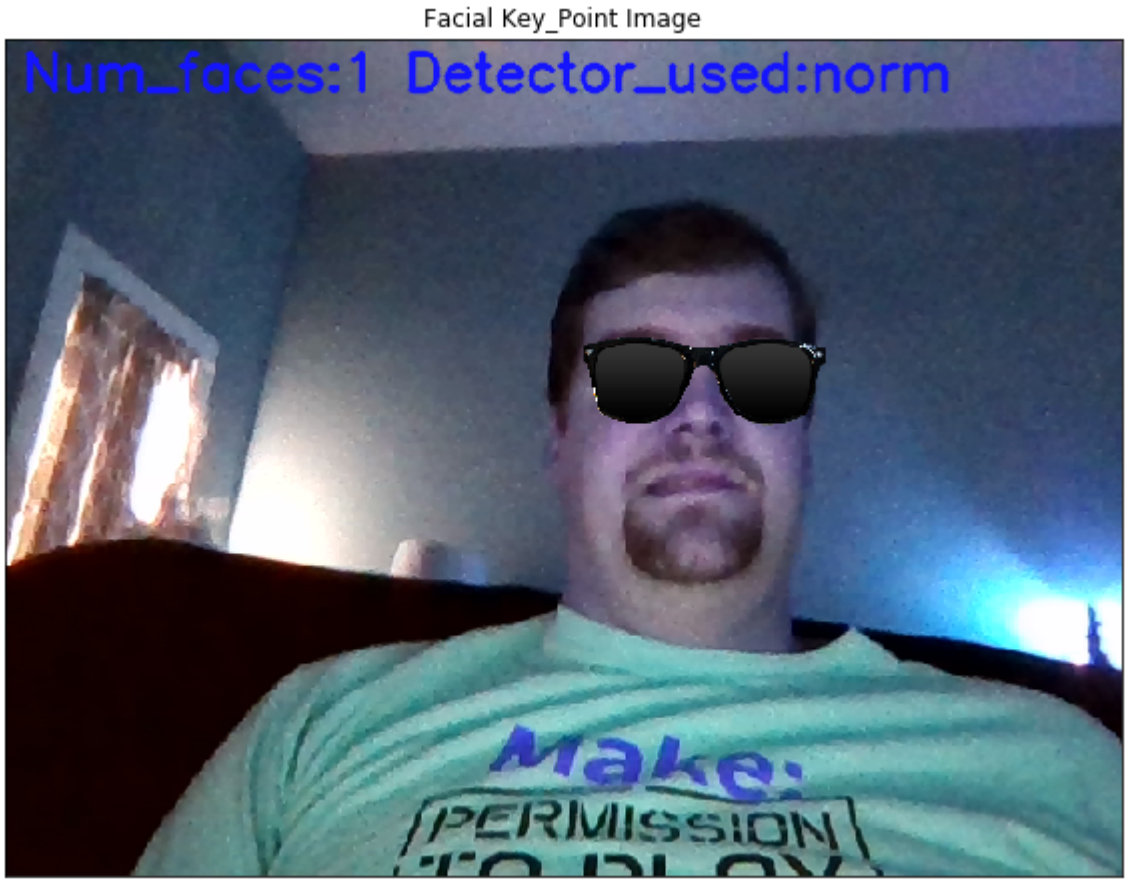
The next Python cell contains the basic laptop video camera function used in the previous optional video exercises. Combine it with the functionality you developed for adding sunglasses to someone's face in the previous optional exercise and you should be good to go!

In [352]:
```
import cv2
import time
from keras.models import load_model
import numpy as np

def laptop_camera_go():
    # Create instance of video capturer
    cv2.namedWindow("face detection activated")
    vc = cv2.VideoCapture(0)

    # try to get the first frame
    if vc.isOpened():
        rval, frame = vc.read()
    else:
        rval = False

    # Keep video stream open
    while rval:
        # Plot image from camera with detections marked
        cv2.imshow("face detection activated", detect_faces_add_glasses(frame
))

        # Exit functionality - press any key to exit laptop video
        key = cv2.waitKey(20)
        if key > 0: # exit by pressing any key
            # Destroy windows
            cv2.destroyAllWindows()

            for i in range (1,5):
                cv2.waitKey(1)
            return detect_faces_add_glasses(frame)

        # Read next frame
        time.sleep(0.05)             # control framerate for computation - def
ault 20 frames per sec
        rval, frame = vc.read()
```

In [373]:
```
face_cascade = cv2.CascadeClassifier('detector_architectures/haarcascade_frontalface_default.xml')

# Run sunglasses painter
frame=laptop_camera_go()

# Display the image
fig = plt.figure(figsize = (10,10))
ax1 = fig.add_subplot(111)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.set_title('Facial Key_Point Image')
ax1.imshow(frame)
```

```
C:\Users\llathrop\AppData\Local\conda\conda\envs\aind-dog\lib\site-packages\ipykernel\__main__.py:21: VisibleDeprecationWarning: using a non-integer number instead of an integer will result in an error in the future
```

Out[373]: <matplotlib.image.AxesImage at 0x205c1dd1438>



possible enhancment would be to warp the glass image to rotate with the face, keyed to the outer eyebrow points.

Submitted for review by Udacity, Dec 4th 2017