In [1]:

```
using CSV, DataFrames, RDatasets, StatsBase
#wipro = CSV.read("C:\\Users\\hp\\.julia\\WIPRO.CSV")
using Pkg
Pkg.add("Images")

import Pkg
Pkg.add("ImageIO")

import Pkg
Pkg.add("Colors")
```

```
   Updating registry at `C:\Users\hp\.julia\registries\General`
  Resolving package versions...
No Changes to `C:\Users\hp\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\hp\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\hp\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\hp\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\hp\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\hp\.julia\environments\v1.5\Manifest.toml`
```

In [2]:

```
import Pkg
Pkg.add("ImageMagick")
```

```
  Resolving package versions...
No Changes to `C:\Users\hp\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\hp\.julia\environments\v1.5\Manifest.toml`
```

In [4]:

```
using CSV, DataFrames, RDatasets, StatsBase
using Images
using ImageIO
using ImageMagick
using Colors
using CSV, DataFrames, RDatasets, StatsBase
using Images
using ImageIO
```

Reading the Training Dataset File

In [5]:

```
train= CSV.read("C:/Users/hp/.julia/Character_Recognition/train_character.CSV");
```

Checking for Missing Values by using the function "describe"

In [6]:

```
describe(train)
```

Out[6]:

4 rows × 8 columns (omitted printing of 5 columns)

	variable	mean	min
	Symbol	Union...	Any
1	path		C:/Users/hp/.julia/Character_Recognition/Training_Dataset/v2-00016.png
2	symbol_id	42.205	31
3	latex		A
4	user_id	9287.19	10

Checking the type of Data

In [7]:

```
println("Data Type: \t\t\t\t", typeof(train))
println("The number of images in the dataset: \t", size(train,1))
```

```
Data Type:                              DataFrame
The number of images in the dataset:    1893
```

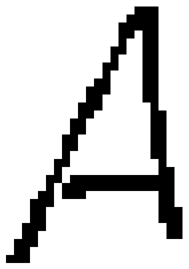
Displaying a few images of the Dataset

In [8]:

```
path=train[1,1]
println(path)
img=load(path)
```

```
C:/Users/hp/.julia/Character_Recognition/Training_Dataset/v2-00016.png
```

Out[8]:

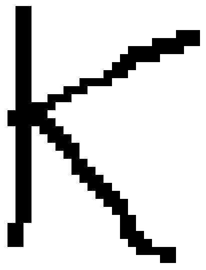


In [15]:

```
path=train[900,1]
println(path)
img=load(path)
```

```
C:/Users/hp/.julia/Character_Recognition/Training_Dataset/v2-17962.png
```

Out[15]:

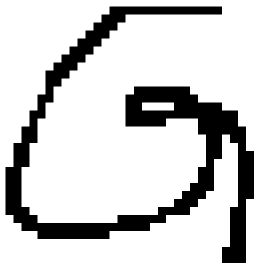


In [10]:

```
path=train[500,1]
println(path)
img=load(path)
```

```
C:/Users/hp/.julia/Character_Recognition/Training_Dataset/v2-17562.png
```

Out[10]:



Datatype and Size of the Image

In [16]:

```
row,col = size(img)
println("The Datatype of the image file is: \t", typeof(img))
println("The Dimensions of the image file is: \t", row, " x ", col)
```

```
The Datatype of the image file is:      Array{RGB{Normed{UInt8,8}},2}
The Dimensions of the image file is:    32 x 32
```

Converting the image to Float64 Datatype

Conversion of datatype of one image (RGB to Gray to Float64)

In [17]:

```
path=train[1,1]
img=load(path)
img = Gray.(img)
img = convert(Array{Float64}, img)
println("The new Datatype is: ", typeof(img))
a,b=size(img)
```

```
The new Datatype is: Array{Float64,2}
```

Out[17]:

```
(32, 32)
```

Conversion of Matrix to Column Vector

In [22]:

```
img = reshape(img, :)
train_im=img
length(train_im)
typeof(train_im)
```

Out[22]:

```
Array{Float64,1}
```

Converting all the images to column vectors and storing them in a single variable

In [23]:

```
for i in 2:size(train,1)
    path=train[i,1]
    img=load(path)
    img = Gray.(img)
    img = convert(Array{Float64}, img)
    img = reshape(img, :)
    train_im=hcat(train_im, img)

end
```

Dimension of the Training Dataset

In [24]:

```
size(train_im)
```

Out[24]:

```
(1024, 1893)
```

Preparing Onehotbatch (Binary Description of the Expected Output)

In [70]:

```
using Flux: onehotbatch, onecold, crossentropy, throttle
```

In [41]:

```
char = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T',
'U','V','W','X','Y','Z']
hotbatch = onehotbatch(char, 'A':'Z')
```

Out[41]:

```
26×26 Flux.OneHotMatrix{Array{Flux.OneHotVector,1}}:
 1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1  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  1
```

Testing Onecold conversion from Binary Input

In [42]:

```
rand_cold=hotbatch[25,:]
typeof(rand_cold)
```

Out[42]:

```
Array{Bool,1}
```

Converting the Boolean Input of Onehotbatch to Float for Onecold Conversion

In [49]:

```
cold_float=convert(Array{Float32}, rand_cold)
cold_float
```

Out[49]:

```
26-element Array{Float32,1}:
 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
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 1.0
 0.0
```

In [50]:

```
char[onecold(cold_float)]
```

Out[50]:

```
'Y': ASCII/Unicode U+0059 (category Lu: Letter, uppercase)
```

Checking the Datatype of the Character Labels

In [51]:

```
character = train[:,3]
typeof(character[1])
```

Out[51]:

```
String
```

Converting String Datatype to Character Datatype

In [55]:

```
char_var =collect(character[1])[1]
typeof(char_var)
```

Out[55]:

```
Char
```

In [56]:

```
for i in 2:size(train,1)
    strchar = collect(character[i])[1]
    char_var=vcat(char_var,strchar)
end
char_var
```

In [60]:

```
println("The number of labels in the dataset is: ", length(char_var))
println("Datatype of the label dataset is: \t", typeof(char))
```

```
The number of labels in the dataset is: 1893
Datatype of the label dataset is: 	Array{Char,1}
```

Onehotbatch Conversion of the Characters in the Training Dataset

In [135]:

```
train_label = onehotbatch(char_var, 'A':'Z')
```

Out[135]:

```
26×1893 Flux.OneHotMatrix{Array{Flux.OneHotVector,1}}:
 1  1  1  1  1  1  0  0  0  0  0  1  1  …  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  1  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  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  1  1  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  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  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  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  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  1  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  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  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  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  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  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  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  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  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  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  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  1  0  0  …  1  1  1  1  1  1  1  1  1  1  1  1
```

In [87]:

```
#import Pkg
#Pkg.add("Base")
using Base.Iterators: repeated
using Flux, Statistics
```

In [120]:

```
layers = Chain(
    Dense(1024, 512, σ),
    Dense(512, 78, σ),
    Dense(78, 26),
    softmax)
```

Out[120]:

```
Chain(Dense(1024, 512, σ), Dense(512, 78, σ), Dense(78, 26), softmax)
```

In [118]:

```
loss(x, y) = crossentropy(layers(x), y)
```

Out[118]:

```
loss (generic function with 1 method)
```

In [138]:

```
dataset = repeated((train_im, train_label), 25)
```

Out[138]:

```
Base.Iterators.Take{Base.Iterators.Repeated{Tuple{Array{Float64,2},Flux.OneHo
tMatrix{Array{Flux.OneHotVector,1}}}}}(Base.Iterators.Repeated{Tuple{Array{Fl
oat64,2},Flux.OneHotMatrix{Array{Flux.OneHotVector,1}}}}(([1.0 1.0 … 1.0 1.0;
1.0 1.0 … 1.0 1.0; … ; 1.0 1.0 … 1.0 1.0; 1.0 1.0 … 1.0 1.0], Bool[1 1 … 0 0;
0 0 … 0 0; … ; 0 0 … 0 0; 0 0 … 1 1])), 25)
```

In [116]:

```
opt = ADAM()
```

Out[116]:

```
ADAM(0.001, (0.9, 0.999), IdDict{Any,Any}())
```

In [140]:

```
evalcb = () -> @show(loss(train_im, train_label))
```

Out[140]:

```
#3 (generic function with 1 method)
```

In [141]:

```
Flux.train!(loss, params(layers), dataset, opt, cb = throttle(evalcb, 10))
```

```
loss(train_im, train_label) = 3.1911008f0
loss(train_im, train_label) = 3.0444183f0
loss(train_im, train_label) = 2.778135f0
```

In [146]:

```
accuracy(x, y) = mean(onecold((layers(x))) .== onecold(y))
```

Out[146]:

```
accuracy (generic function with 1 method)
```

In [147]:

```
@show accuracy(train_im, train_label)
```

```
accuracy(train_im, train_label) = 0.36397253037506605
```

Out[147]:

```
0.36397253037506605
```

In [169]:

```
dataset = repeated((train_im, train_label), 100)
```

Out[169]:

```
Base.Iterators.Take{Base.Iterators.Repeated{Tuple{Array{Float64,2},Flux.OneHo
tMatrix{Array{Flux.OneHotVector,1}}}}}(Base.Iterators.Repeated{Tuple{Array{Fl
oat64,2},Flux.OneHotMatrix{Array{Flux.OneHotVector,1}}}}(([1.0 1.0 … 1.0 1.0;
1.0 1.0 … 1.0 1.0; … ; 1.0 1.0 … 1.0 1.0; 1.0 1.0 … 1.0 1.0], Bool[1 1 … 0 0;
0 0 … 0 0; … ; 0 0 … 0 0; 0 0 … 1 1])), 100)
```

In [170]:

```
Flux.train!(loss, params(layers), dataset, opt, cb = throttle(evalcb, 10))
```

```
loss(train_im, train_label) = 0.30629227f0
loss(train_im, train_label) = 0.24503887f0
loss(train_im, train_label) = 0.19712085f0
loss(train_im, train_label) = 0.1589241f0
loss(train_im, train_label) = 0.12828606f0
loss(train_im, train_label) = 0.101597756f0
loss(train_im, train_label) = 0.082155965f0
loss(train_im, train_label) = 0.0667491f0
loss(train_im, train_label) = 0.0548043f0
```

In [171]:

```
@show accuracy(train_im, train_label)
```

```
accuracy(train_im, train_label) = 0.9984152139461173
```

Out[171]:

```
0.9984152139461173
```

In [157]:

```
dataset = repeated((train_im, train_label), 150)
```

Out[157]:

```
Base.Iterators.Take{Base.Iterators.Repeated{Tuple{Array{Float64,2},Flux.OneHo
tMatrix{Array{Flux.OneHotVector,1}}}}}(Base.Iterators.Repeated{Tuple{Array{Fl
oat64,2},Flux.OneHotMatrix{Array{Flux.OneHotVector,1}}}}(([1.0 1.0 … 1.0 1.0;
1.0 1.0 … 1.0 1.0; … ; 1.0 1.0 … 1.0 1.0; 1.0 1.0 … 1.0 1.0], Bool[1 1 … 0 0;
0 0 … 0 0; … ; 0 0 … 0 0; 0 0 … 1 1])), 150)
```

In [153]:

```
Flux.train!(loss, params(layers), dataset, opt, cb = throttle(evalcb, 10))
```

```
loss(train_im, train_label) = 0.55345404f0
loss(train_im, train_label) = 0.47783762f0
loss(train_im, train_label) = 0.408552f0
loss(train_im, train_label) = 0.35090527f0
loss(train_im, train_label) = 0.30668303f0
loss(train_im, train_label) = 0.26540643f0
loss(train_im, train_label) = 0.23347075f0
loss(train_im, train_label) = 0.20347238f0
loss(train_im, train_label) = 0.17799254f0
loss(train_im, train_label) = 0.15631826f0
loss(train_im, train_label) = 0.13787028f0
loss(train_im, train_label) = 0.1234828f0
loss(train_im, train_label) = 0.10987445f0
loss(train_im, train_label) = 0.098223135f0
```

In [156]:

```
@show accuracy(train_im, train_label)
```

```
accuracy(train_im, train_label) = 0.9978869519281564
```

Out[156]:

```
0.9978869519281564
```

In [163]:

```
layers = Chain(
    Dense(1024, 512, relu),
    Dense(512, 78, relu),
    Dense(78, 26),
    softmax)
```

Out[163]:

```
Chain(Dense(1024, 512, relu), Dense(512, 78, relu), Dense(78, 26), softmax)
```

In [165]:

```
dataset = repeated((train_im, train_label), 100)
```

Out[165]:

```
Base.Iterators.Take{Base.Iterators.Repeated{Tuple{Array{Float64,2},Flux.OneHo
tMatrix{Array{Flux.OneHotVector,1}}}}}(Base.Iterators.Repeated{Tuple{Array{Fl
oat64,2},Flux.OneHotMatrix{Array{Flux.OneHotVector,1}}}}(([1.0 1.0 … 1.0 1.0;
1.0 1.0 … 1.0 1.0; … ; 1.0 1.0 … 1.0 1.0; 1.0 1.0 … 1.0 1.0], Bool[1 1 … 0 0;
0 0 … 0 0; … ; 0 0 … 0 0; 0 0 … 1 1])), 100)
```

In [166]:

```
Flux.train!(loss, params(layers), dataset, opt, cb = throttle(evalcb, 10))
```

```
loss(train_im, train_label) = 3.5547488f0
loss(train_im, train_label) = 2.7925591f0
loss(train_im, train_label) = 2.197669f0
loss(train_im, train_label) = 1.6558882f0
loss(train_im, train_label) = 1.2180454f0
loss(train_im, train_label) = 0.8899417f0
loss(train_im, train_label) = 0.645422f0
loss(train_im, train_label) = 0.50560296f0
loss(train_im, train_label) = 0.394651f0
loss(train_im, train_label) = 0.3126844f0
```

In [167]:

```
@show accuracy(train_im, train_label)
```

```
accuracy(train_im, train_label) = 0.9413629160063391
```

Out[167]:

```
0.9413629160063391
```

In [ ]:

```

```