

## ML Assignment – 3 Logistic Regression and SVM

**Participants:** Keshav Narayan Srinivasan (50610509)

Pramila Yadav (50613803)

Hari Chandan Gooda (50614165)

### Implement Logistic Regression and give the prediction results

#### Feature Selection Detail:

To optimize model training, we are ignoring features which remain same across all samples because it will not provide any learning value to training set and we are selecting features which will provide meaningful insights.

Total no. of selected features is 717

```
12 13 14 15 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 86 87
88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129
130 131 132 133 134 135 136 137 138 139 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 169 1
70 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 20
7 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244
245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281
282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 3
19 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 35
6 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393
394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430
431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 4
68 469 470 471 472 473 474 475 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 50
6 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543
544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581
582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 6
19 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 646 647 648 649 650 651 652 653 654 655 656 657 65
8 659 660 661 662 663 664 665 666 667 668 669 670 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698
702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 731 732 733 734 735 736 737 738 739 740 741 742
743 744 745 746 747 748 749 750 751 752 753 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779
Total number of selected features : 717
```

#### Use the Support Vector Machine (SVM) to perform classification:

For optimizing the neural network's hyperparameters focused on two key elements which are regularization coefficient ( $\lambda$ ) and the number of hidden units ( $m$ ).

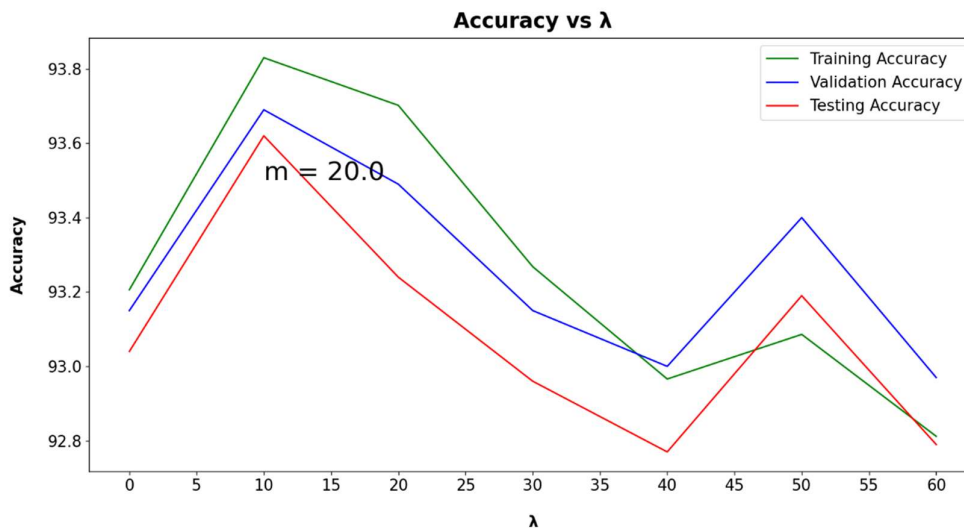
To address **overfitting**, we applied regularization and found that the number of hidden units significantly affects the model's performance. We adjusted  $m$  value from 4 to 20 (step size 4) and  $\lambda$  value from 0 to 60 (step size 10)

Lambda	Training Accuracy	Validation Accuracy	Test Accuracy	m
10	93.83	93.69	93.62	20
20	93.702	93.49	93.24	20
50	93.086	93.4	93.19	20
0	93.206	93.15	93.04	20
30	93.268	93.15	92.96	20
10	93.346	93.09	92.91	16
60	92.812	92.97	92.79	20
40	92.966	93	92.77	20
40	92.41	92.1	92.45	12
40	92.39	92.4	92.4	16
0	92.774	92.64	92.37	16
30	92.528	92.55	92.35	16
20	92.812	92.74	92.32	16

From the above table we can get optimum Lambda as 10 &  $m$  as 20 giving Test Accuracy of 93.62%.

Taking  $m$  value as 20

Lambda	Training Accuracy	Validation Accuracy	Test Accuracy	m
10	93.83	93.69	93.62	20
20	93.702	93.49	93.24	20
50	93.086	93.4	93.19	20
0	93.206	93.15	93.04	20
30	93.268	93.15	92.96	20
60	92.812	92.97	92.79	20
40	92.966	93	92.77	20



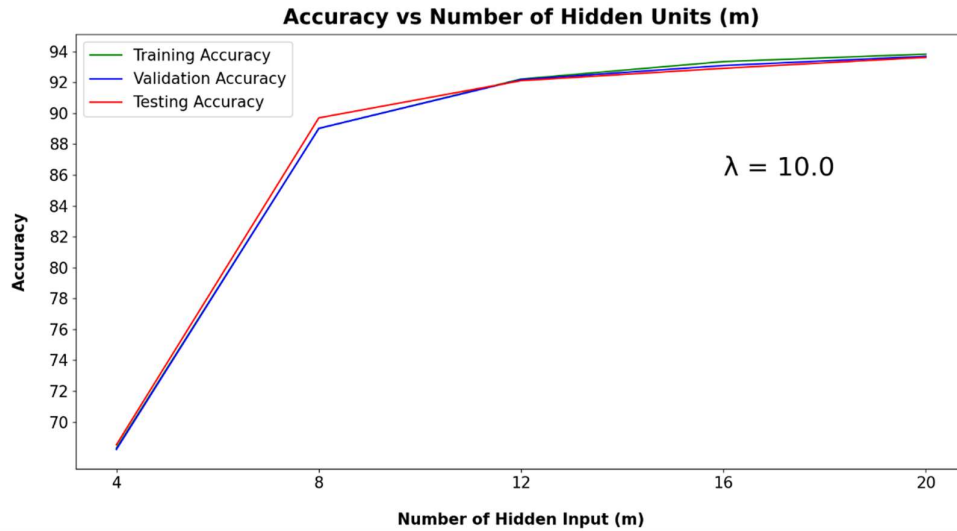
The highest accuracy in testing was observed when  $\lambda$  is set to 10. As  $\lambda$  increases, there are slight variations in training, testing and validation datasets. According to theory, accuracy should decrease after reaching the optimal point due to underfitting, and our plot shows the similar behaviour. There are some minor anomalies occur, which are justifiable in a practical setup, with similar inconsistencies seen across the training, testing and validation datasets.

Taking  $\lambda$  as 10 and explore different values for m.

Taking  $\lambda$  value as 10

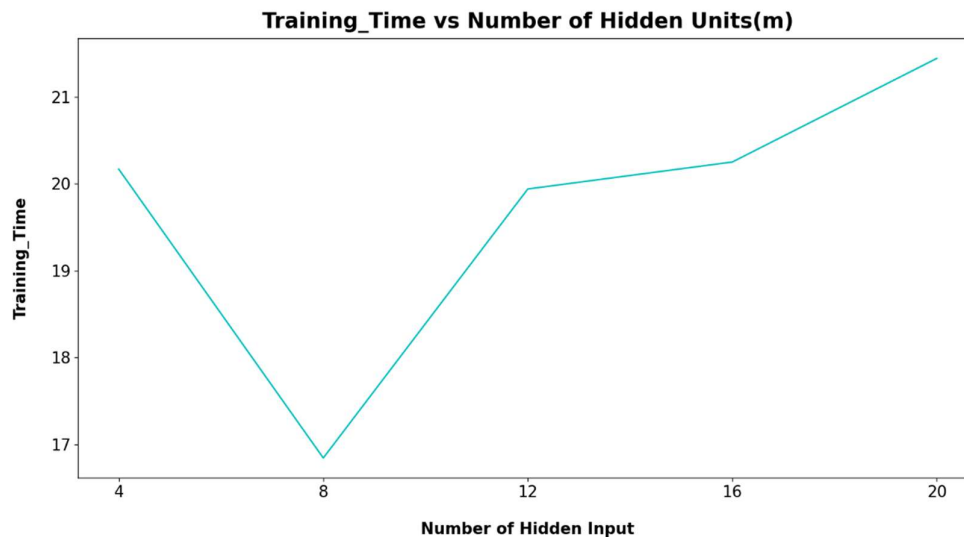
10	68.32	68.23	68.53	4
10	89.012	89.02	89.7	8
10	92.216	92.18	92.11	12
10	93.346	93.09	92.91	16
10	93.83	93.69	93.62	20

From above graph it is evident that when m is 20, we will get best result.



The plot shows that as  $m$  grows, accuracy also improves. There is a significant jump between 4 and 8, after which the increase is slow and the curve levels off from 16 to 20. This indicates that after a certain point, adding more hidden units does not significantly impact accuracy.

Plotting Training time vs no of hidden units:

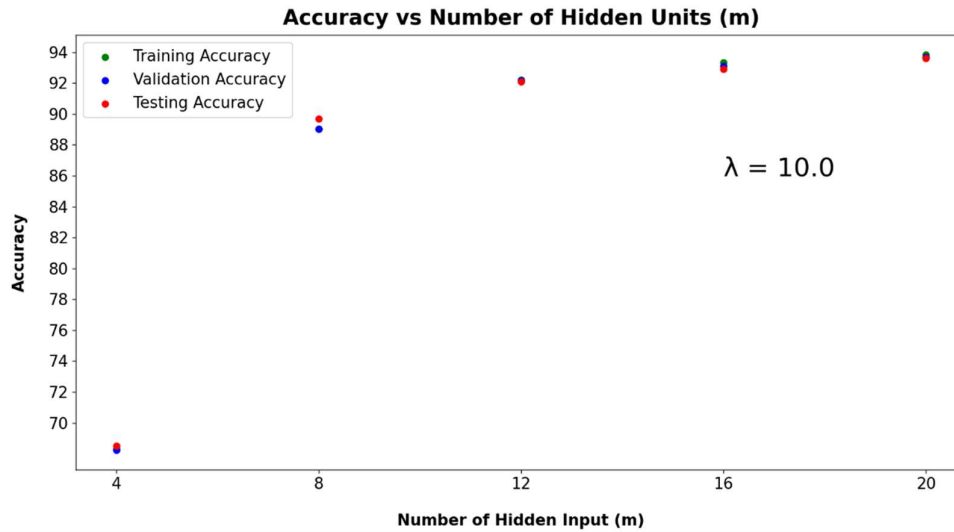


The plot indicates that as we add hidden units, training time increases. So, we should think carefully when to stop adding more units to avoid making the training time longer than needed, especially if accuracy stops improving.

**Classification Accuracy on Handwritten Digits Test Data using nnScript.py:**

Best Hyper Parameter combination:  $\lambda = 10$ ,  $m = 20$

scatter plot of accuracy v/s  $m$



### Classification accuracy on the celebA dataset:

Using implementations of the functions: `sigmoid()`, `nnObjFunc()`, and `nPredict()` from `nnScript` in `facennScript`. Below are the results obtained by running ``facennScript.py`` when  $\lambda$  is set to 10 :

```
Training set Accuracy:84.37440758293839%
Validation set Accuracy:82.92682926829268%
Test set Accuracy:84.21650264950796%
```

### Compare the results of deep neural network and neural network with one hidden layer on the CelebA data set:

We analysed and compared the performance on the celebA dataset between a single-layer neural network and a multilayer deep neural network. By adjusting the layers from 2 to 7 (2, 3, 5, 7), we obtained the following results:

No of Layers	Script	Accuracy in %	Training Time in sec	Avg loss
1	facennScript.py	86.26	48.068	
2	deepnnScript.py	67.1	44.47	0.662335
3	deepnnScript.py	64.1	50.36	0.687593
5	deepnnScript.py	50.0	55.47	0.693011
7	deepnnScript.py	50.0	62.22	0.693166

### Observation:

Adding more hidden layers makes the training time longer because the model becomes more complex when we add more hidden layers.

We also found that accuracy goes down with more layers. This might be because of overfitting, which causes good performance on training data but worse performance on test data.

## CNN Results: Accuracy and Training Duration Analysis:

In term of Accuracy Comparison, The CNN model's accuracy is 98.7%, which is better than a simple or multilayer neural network

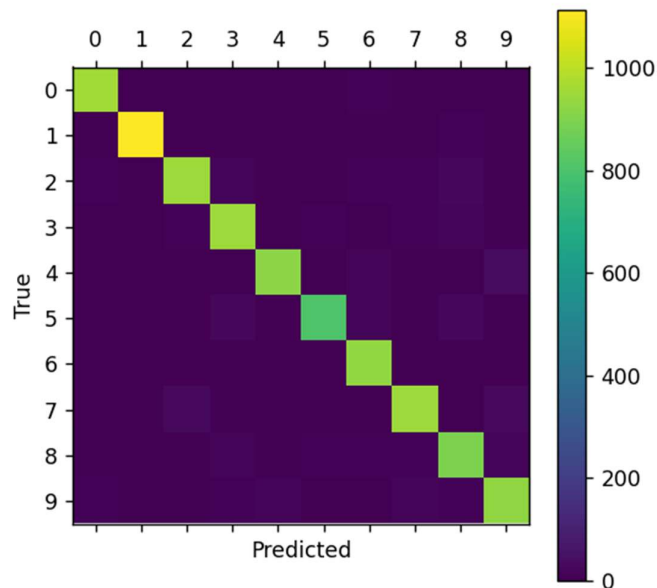
The reason why CNN Works Well is that CNN adapts its settings to match what the image needs, also it doesn't need prior knowledge to pick important features.

The Importance of Filters in CNN help break down the image into smaller sections and analysed each part which improves the accuracy.

The Training Time is 16 seconds for the CNN model using CUDA threading to achieve 93.3% accuracy.

```
loss: 0.461491 [44800/60000]
loss: 0.173865 [51200/60000]
loss: 0.283974 [57600/60000]
Time usage: 0:00:16
Test Error:
Accuracy: 93.3%, Avg loss: 0.217007

Example errors:
Confusion Matrix:
[[ 956  0  0  1  0  4 12  2  5  0]
 [  0 1109  3  2  1  1  5  0 14  0]
 [ 12  1 916 17 13  2 18 18 31  4]
 [  3  5  9 930  0 20  1 15 18  9]
 [  0  3  2  0 918  0 18  1  2 38]
 [  6  2  1 14  6 823 18  3 14  5]
 [  6  4  1  0  9 13 922  2  1  0]
 [  0  9 20  5  4  0  0 935  3 52]
 [  8  4  3 13 11 12 11  9 884 19]
 [  8  5  3  9 23  4  1 13  6 937]]
```



The matrix shows that nearly all predictions are correct resulting in a final accuracy of 93.3% and it has few mistakes.