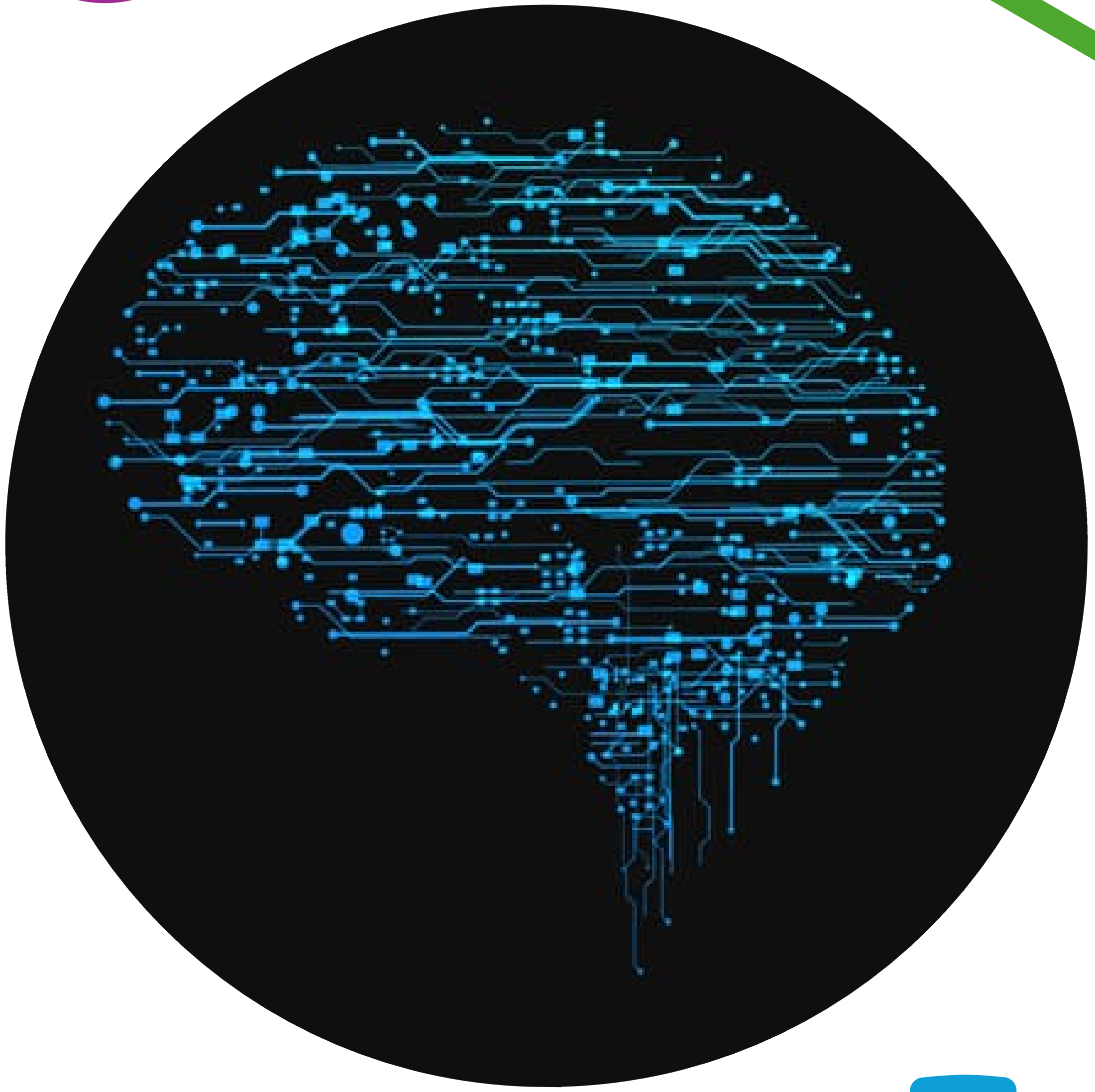


Possible applications of neural networks in non-life reserving

Aktuárský seminář MFF UK 12.12.2025

RNDr. Vojtěch Bednařík

About me

- Graduate of MFF UK
- Started at EY as a consultant in September 2015
- Current position: an actuarial manager in Risk Consulting
- Focus on Non-Life reserving
- Experience from consulting projects and audit support

Shape the future
with confidence

Agenda

1. Short introduction to neural networks
2. Obtaining data from a publicly available generator for individual claims
3. Data analysis and application of chain ladder
4. Presentation of a simple neural network model based on chain ladder approach
5. Application of the model in R
6. Results - comparison to chain ladder result and true values
7. Stability considerations
8. Pros and cons of the model
9. Overview of other explorations in neural networks models in non-life reserving
10. Key ideas applied in neural network models in non-life reserving
11. Conclusion

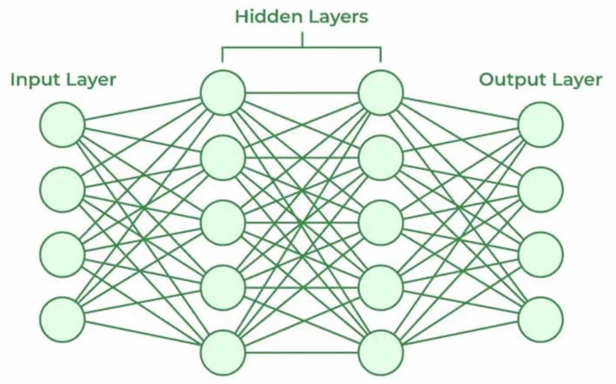
Motivation

Chain ladder is a well-established method in non-life reserving

With more data and computation capacity being available, is there a method, which could benefit from this?

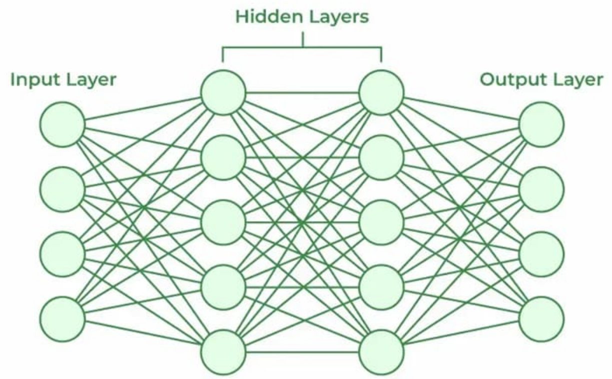
1. Short introduction to neural networks

What are neural networks?



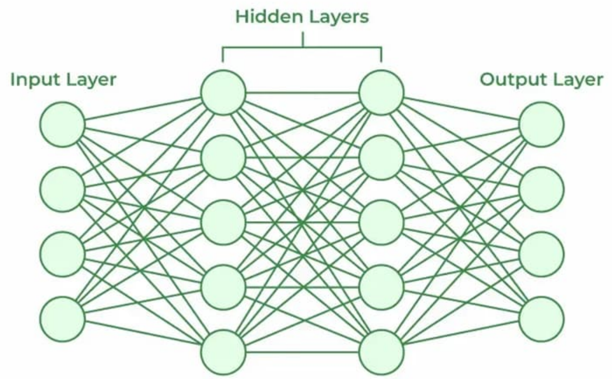
- Neural networks are learning models that mimic complex functions of human brain
- No statistical assumptions are stated like in generalized linear regression, assumptions are only made on the structure of models, which consist of:
  - **Neurons**: basic units that receive inputs
  - **Connections**: links between neurons regulated by an activation function, weights and biases
  - **Weights and biases**: parameters determining strength and influence of connections, which is analogue to regression parameters and intercepts

Assumptions of neural networks



- Neurons are arranged in layers, which are sequentially arranged
- Neurons within the same layer do not interact or communicate with each other
- All inputs enter into the network through an input layer and are passed through hidden layers to the output layer
- Each hidden layer has the same activation function
- Neurons at consecutive layers are densely connected
- Each layer has its own weights and biases associated with it

Working of neural networks



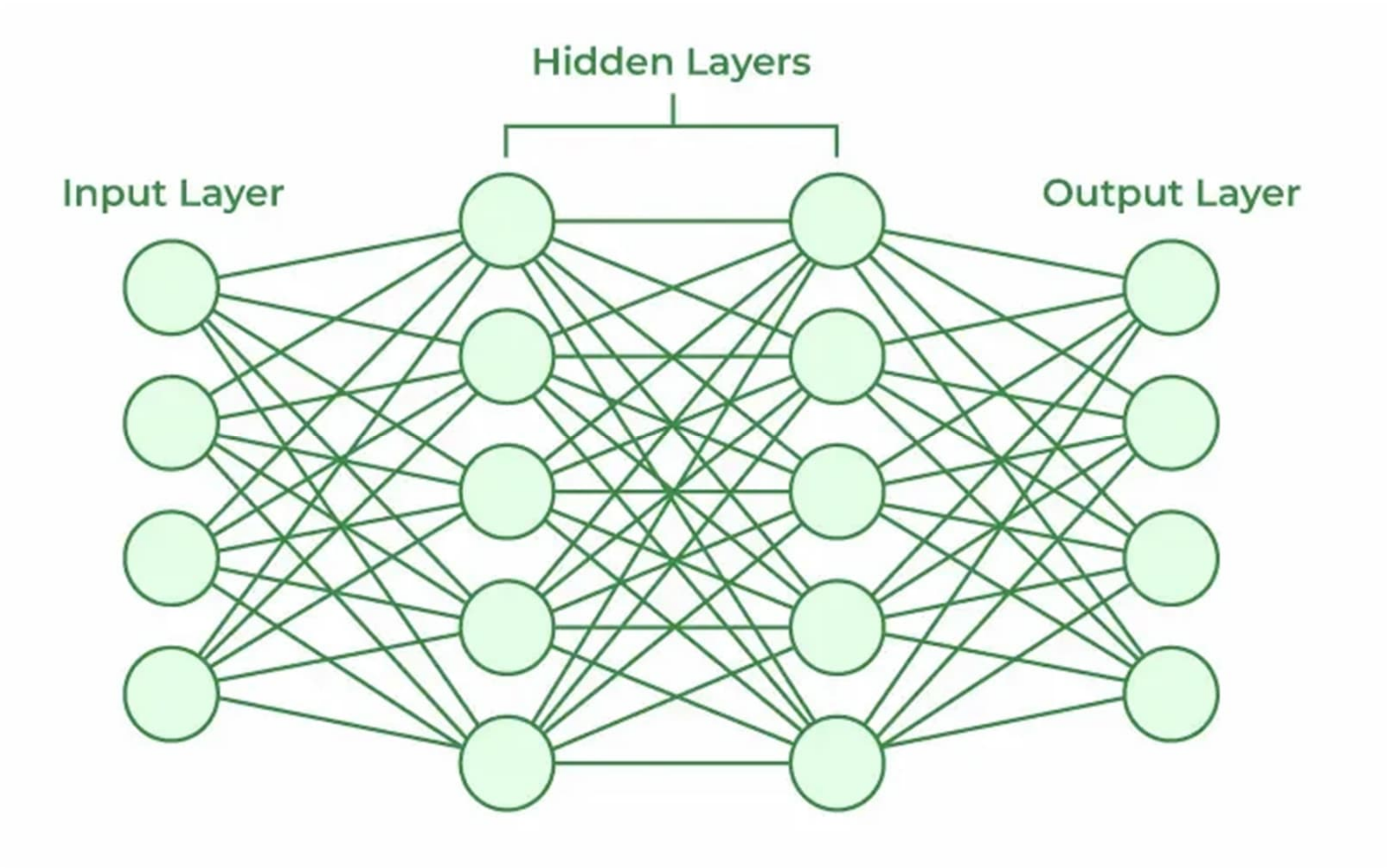
- Available inputs define an input layer, we can follow the example on the picture in the top right corner, so we have inputs x_1, x_2, x_3, x_4
- The first hidden layer is determined by an activation function f_1, weights and biases, for example k-th neuron is calculated as

$$z_k = f_1\left(w_{k,0} + w_{k,1}x_1 + w_{k,2}x_2 + w_{k,3}x_3 + w_{k,4}x_4\right) \text{ for } k = 1, \dots, 5$$

- The second hidden layer has a different activation function f_2 and different weights and biases, the first hidden layer works here as an input layer (inputs are z_1, z_2, z_3, z_4, z_5)
- Finally, an output layer works with different weights and biases, the second hidden layer is an input, each neuron can have its own activation function

Hyperparameters and choices

- Number of hidden layers
- Numbers of neurons
- Activation functions
- Split to training and validation sets
- Loss function for minimization
- Algorithm for minimization
- Number of epochs
- Batch sizes
- Learning rate

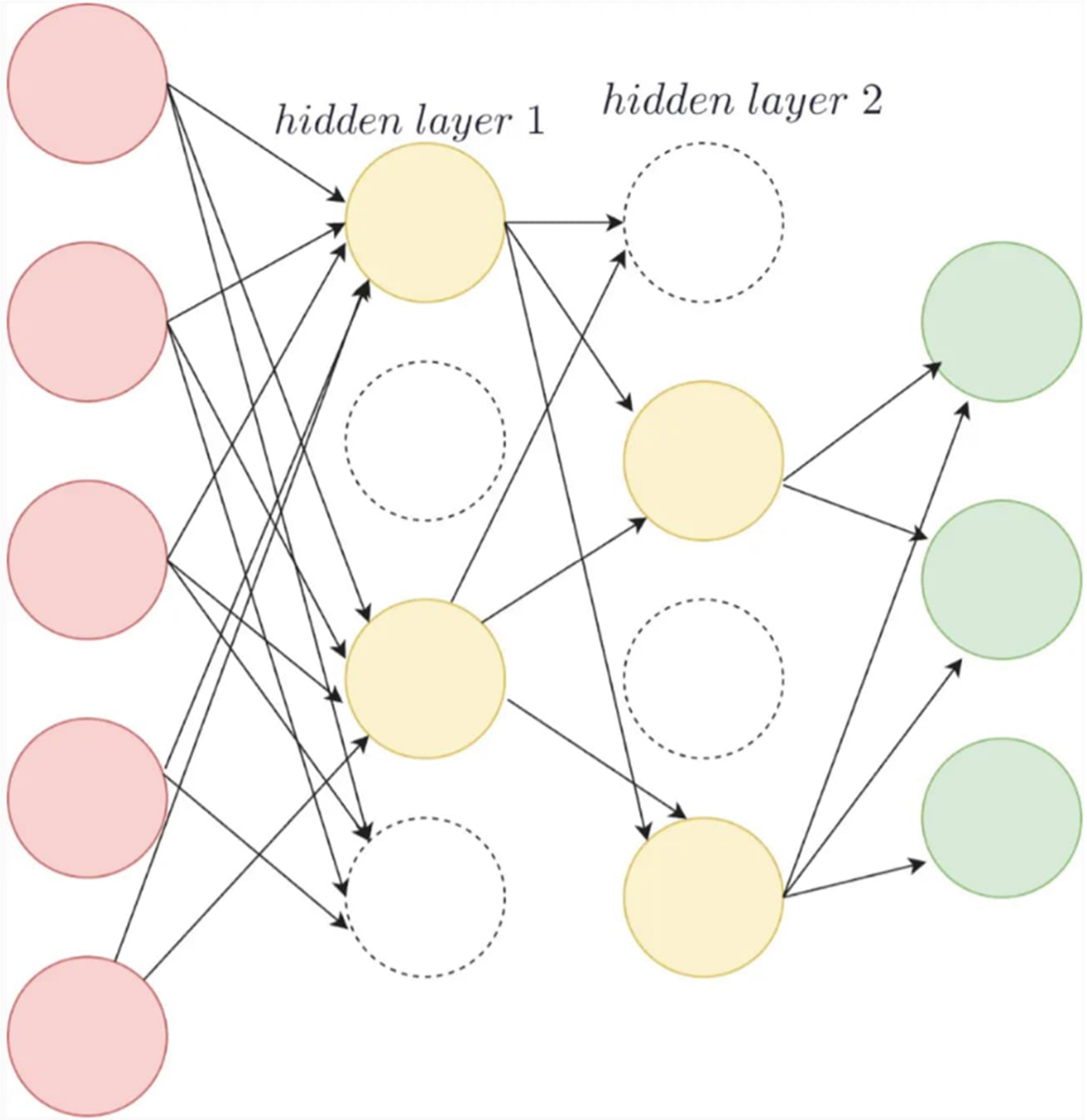


Regularization techniques (examples)

- Early stopping
- L1 and L2 regularization
- Addition of noise
- Dropout

Dropout

- Dropout means an additional layer, which is appended to a layer in a neural network
- In each step, when parameters are updated, some neurons are ignored (set equal to zero)
- Neurons excluded from updates are selected randomly with a given probability p (i.e., each neuron is excluded with probability p)
- Some rescaling is necessary
- Recommended probability is usually in range 20 - 50 %



2. Obtaining data from a publicly available generator for individual claims

Description of the generator

- Source: [1]
- A user-friendly R script is publicly available
- Data generator was calibrated on real insurance data
- The generator utilizes a sophisticated system of neural networks
- Users choose following parameters:
  - Two random seeds
  - Total number of claims
  - 4 growth parameter of portfolios and split of claims
  - Volatility total claim sizes and recoveries
- Output provides data with following features:
  - Claim number (ClNr)
  - Portfolio (LoB)
  - Accident year (AY), accident quarter (AQ)
  - cc, age, inj_part
  - Reporting delay (RepDel)
  - Payments (Pay00, Pay01, ..., Pay11)
  - Indicator if claims are open or close (Open00, Open01, ..., Open11)

```r
#############################
### Generate the features ###
#############################

V <- 5000000                               # totally expected number of claims (over 12 accounting years)
LoB.dist <- c(0.20,0.40,0.15,0.25)         # categorical distribution for the allocation of the claims to the 4 lines of business
inflation <- c(0.20,0.30,0.05,0)           # growth parameters (per LoB) for the numbers of claims in the 12 accident years
seed1 <-  2212                             # setting seed for simulation
features <- Feature.Generation(V = V, LoB.dist = LoB.dist, inflation = inflation, seed1 = seed1)

str(features)
summary(features)

##############################################
### Simulate (and store) cash flow patterns ###
##############################################

npb <- nrow(features)                     # blocks for parallel computing
seed1 <- 2212                             # setting seed for simulation
std1 <- 1.20                              # standard deviation parameter for total claim size simulation
std2 <- 0.20                              # standard deviation parameter for recovery simulation
output <- Simulation.Machine(features = features, npb = npb, seed1 = seed1, std1 = std1, std2 = std2)
```

Description of data

- ClNr: a unique identifier for each claim simply labeled 1, 2, 3, ...
- LoB: categories from 1 to 4
- cc: claim code categories from 1 to 53
- AY: integer values from 1994 to 2005
- AQ: integer values from 1 to 4
- age: an age of injured person, integer values from 15 to 70
- inj_part: a body part injured, categories from 1 to 99
- RepDel: a reporting delay, integer values from 0 to 11
- Pay00, Pay01, ..., Pay11: (positive or negative) incremental cash flows
- Open00, Open01, Open11: an indicator of open or closed claims, binary values

ClNr	LoB	cc	AY	AQ	age	inj_part	RepDel	Pay00	Pay01	Pay02	Pay03	Pay04	Pay05	Pay06	Pay07	Pay08	Pay09	Pay10	Pay11	Open00	Open01	Open02	Open03	Open04	Open05	Open06	Open07
1	4	6	1994	3	38	33	0	1536	905	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
2	4	43	1994	4	27	12	0	196	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	4	31	1994	3	44	23	0	1311	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	13	1994	3	36	12	0	386	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0
5	4	14	1994	1	33	21	0	1796	802	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
6	4	47	1994	1	20	12	0	373	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	33	1994	1	35	99	0	3574	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	4	50	1994	2	26	36	0	137	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	2	51	1994	4	39	56	0	2010	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
10	4	15	1994	3	30	10	0	2443	1851	901	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Data changes before modelling

- 5 004 000 claim observations of 32 variables available as an output
- Incremental payments Pay00, Pay01, ..., Pay11 transformed to cumulative payments C01, C02, ..., C11
- Incremental payments Pay00, Pay01, ..., Pay11 and indicators Open00, Open01, ..., Open11 are then excluded
- 2 rows with negative claims omitted
- 48 376 claims will be reported in future (these rows are put aside from the data set for neural network training)
- 1 584 484 rows with zero claims excluded, i.e., final count of observations is 3 371 138 claims
- Categorical variables (LoB, cc, AQ, inj_part) transformed into indicators via so-called one-hot encoding, e.g., for cc we create cc1, cc2, ..., cc53, for which holds:

$$cc_k = 1, \text{ if } cc = k$$

$$cc_k = 0, \text{ otherwise}$$

- Age is transformed to a variable in range [-1, 1]:

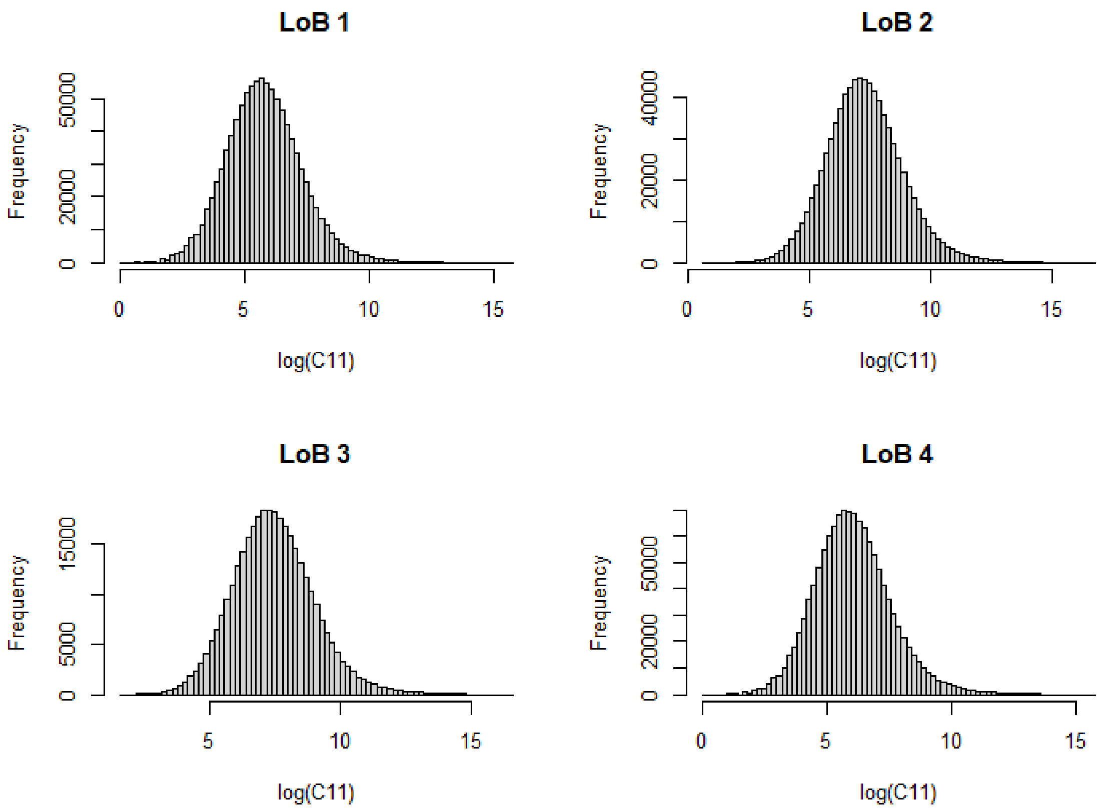
$$age_{transformed} = 2 * \frac{age - \min(age)}{\max(age) - \min(age)} - 1$$

- Inclusion of new variables and exclusion of old variables lead to a dataset with 120 columns

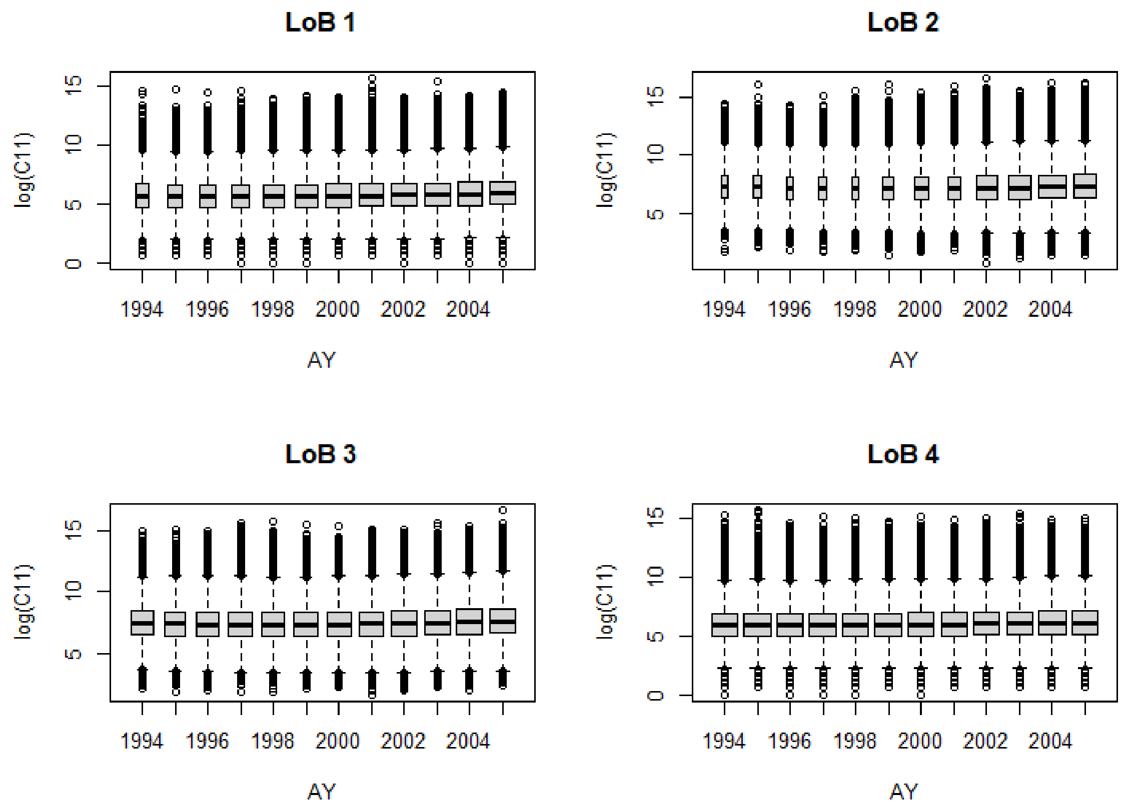
3. Data analysis and application of chain ladder

Analysis of claims (including future development)

Histogram of log ultimate payments



Boxplot of log ultimate payments by accident years



Analysis of claims (including future development)

Mean				
AY	**LoB 1**	**LoB 2**	**LoB 3**	**LoB 4**
1994	1 518	7 537	7 786	2 128
1995	1 427	7 894	7 529	2 402
1996	1 418	6 139	8 338	2 243
1997	1 496	7 019	8 486	2 308
1998	1 483	7 360	8 130	2 286
1999	1 554	7 158	8 347	2 415
2000	1 596	7 275	8 212	2 436
2001	1 695	7 545	9 607	2 588
2002	1 696	8 246	9 621	2 606
2003	1 767	8 363	9 745	2 840
2004	1 750	8 932	10 186	2 906
2005	1 950	9 183	11 818	3 029
Total	**1 655**	**8 354**	**9 119**	**2 515**

Median				
AY	**LoB 1**	**LoB 2**	**LoB 3**	**LoB 4**
1994	296	1 444	1 685	381
1995	293	1 393	1 575	377
1996	293	1 334	1 551	373
1997	290	1 287	1 513	374
1998	292	1 277	1 471	373
1999	294	1 255	1 491	377
2000	299	1 259	1 530	382
2001	302	1 271	1 562	389
2002	313	1 319	1 641	402
2003	323	1 371	1 687	414
2004	340	1 440	1 821	433
2005	350	1 536	1 962	456
Total	**310**	**1 385**	**1 628**	**393**

Standard deviation				
AY	**LoB 1**	**LoB 2**	**LoB 3**	**LoB 4**
1994	18 696	49 140	56 565	26 011
1995	16 198	104 567	49 780	38 358
1996	13 352	38 368	64 185	21 481
1997	15 673	55 740	76 544	25 225
1998	11 547	78 903	69 632	23 204
1999	14 109	80 613	64 622	24 085
2000	13 684	59 704	57 677	24 425
2001	27 091	73 348	72 578	24 330
2002	15 649	86 468	64 623	26 461
2003	20 386	70 992	70 861	34 359
2004	13 705	77 504	71 397	25 272
2005	17 106	79 886	119 367	27 131
Total	**17 222**	**76 594**	**73 266**	**27 107**

Statistics of claim counts

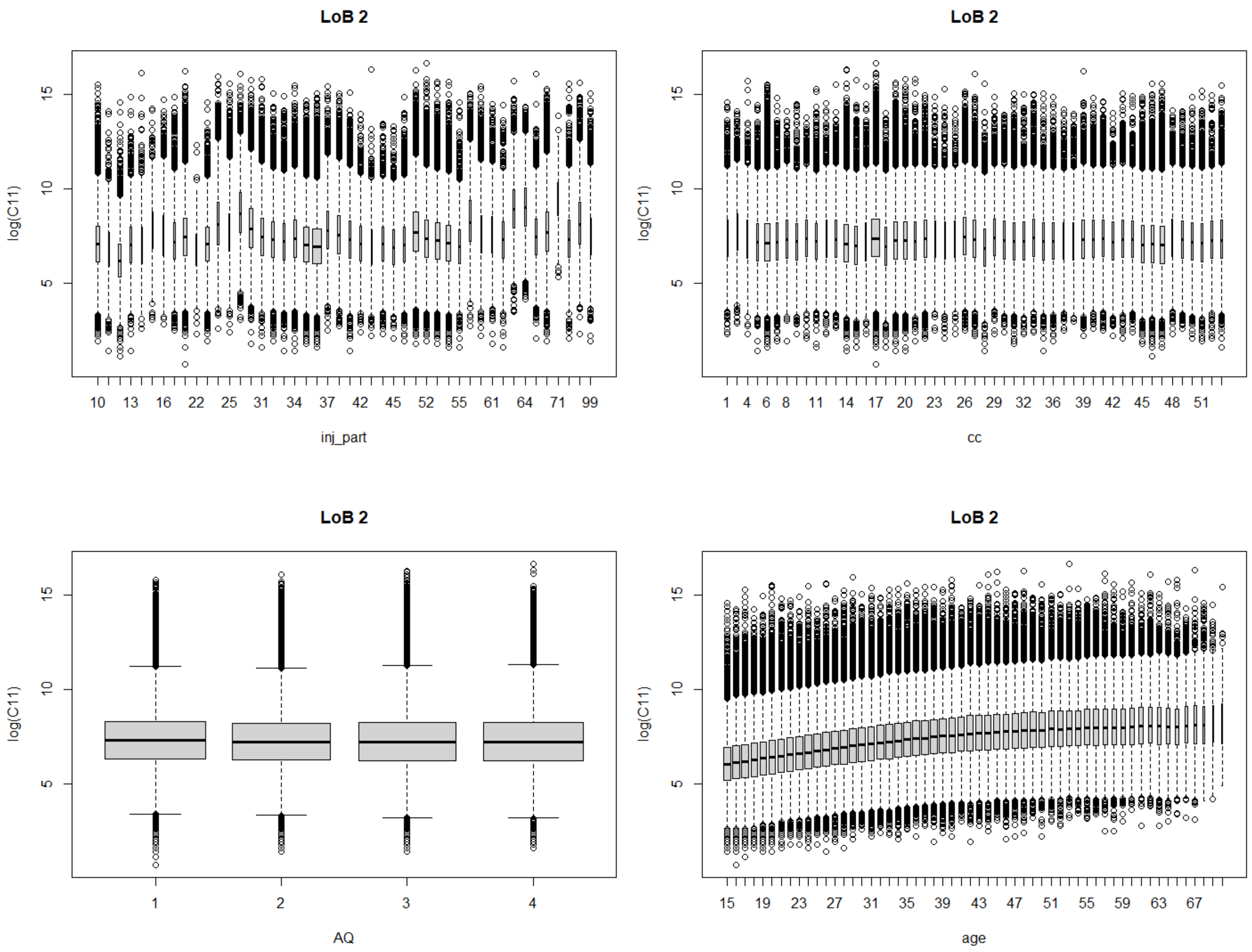
Count of payments	LoB 1	LoB 2	LoB 3	LoB 4
0	6 437	1 179 494	413 869	8 485
1	836 421	756 084	303 578	986 799
2	126 050	45 439	25 100	195 236
3	16 924	11 378	4 292	31 921
4	5 974	4 304	1 572	11 876
5	2 990	1 707	749	5 753
6	1 712	875	417	3 126
7	1 080	515	250	2 016
8	682	371	168	1 301
9	540	280	141	1 036
10	420	272	119	821
11	533	355	164	1 067
12	1 099	313	285	1 610
Total	**1 000 862**	**2 001 387**	**750 704**	**1 251 047**

Accident year	LoB 1	LoB 2	LoB 3	LoB 4
1994	32 779	18 182	49 082	104 270
1995	40 881	28 011	49 901	104 957
1996	45 437	33 444	48 264	104 047
1997	60 413	30 607	56 648	104 273
1998	75 994	37 959	58 273	104 393
1999	87 274	86 346	60 587	104 307
2000	110 659	136 426	61 504	103 888
2001	100 650	126 419	65 793	103 748
2002	120 931	305 388	71 988	104 320
2003	122 090	333 958	77 583	104 720
2004	98 676	498 376	73 411	103 906
2005	105 078	366 271	77 670	104 218
Total	**1 000 862**	**2 001 387**	**750 704**	**1 251 047**

Shares	LoB 1	LoB 2	LoB 3	LoB 4
Number of claims	20,0%	40,0%	15,0%	25,0%
Ultimate claims	11,2%	46,7%	20,9%	21,2%

Detail of a selected LoB

- LoB 2 is the largest in terms of number of claims (including zero claims) and the reserve
- Observable a clear trend in age
- inj_part shows some significant differences across groups
- AQ and cc visually look similar, but the y-axis is in natural logarithmic scale



Chain ladder notation

- I the last accident year
- J the last development year
- Typical case is $I = J$ (we assume this)
- $0 \leq i \leq I$ indexes for accident years
- $0 \leq j \leq J$ indexes for development years
- $C_{i,j}$ cumulative (paid) claims amount of accident year i up to development year j
- $f_0, \ldots, f_{J-1}$ development factors
- $F_{i,0}, \ldots, F_{i,J-1}$ ratios, i.e., $F_{i,j} = \frac{C_{i,j+1}}{C_{i,j}}$
- $D_k = \{C_{i,j}; 0 \leq j \leq k, 0 \leq i + j \leq I\}$ for $0 \leq k \leq J$

	0	1	2	.	.	.	.	.	.	J-2	J-1	J
0	$C_{0,0}$	$C_{0,1}$	$C_{0,2}$	.	.	.	.	.	.	$C_{0,J-2}$	$C_{0,J-1}$	$C_{0,J}$
1	$C_{1,0}$	$C_{1,1}$	$C_{1,2}$	.	.	.	.	.	.	$C_{1,J-2}$	$C_{1,J-1}$	
2	$C_{2,0}$	$C_{2,1}$	$C_{2,2}$	.	.	.	.	.	.	$C_{2,J-2}$		
.	.	.	.	.	.	.	.	.	.			
.	.	.	.	.	.	.	.	.				
.	.	.	.	.	.	.	.					
.	.	.	.	.	.	.						
.	.	.	.	.	.							
.	.	.	.	.								
I-2	$C_{I-2,0}$	$C_{I-2,1}$	$C_{I-2,2}$									
I-1	$C_{I-1,0}$	$C_{I-1,1}$										
I	$C_{I,0}$											

Chain ladder estimate

- Assumptions of chain ladder:
  1. independence of $\{C_{i,j}, 0 \leq j \leq J\}$ and $\{C_{k,j}, 0 \leq j \leq J\}$ for any different accident years $i \neq k$
  2. there exist positive parameters for $0 \leq j < J$ fulfilling
$$\mathrm{E}\left(C_{i,j} | D_{j-1}\right) = f_{j-1} \cdot C_{i,j-1}$$
$$\mathrm{Var}\left(C_{i,j} | D_{j-1}\right) = \sigma_{j-1}^2 \cdot C_{i,j-1}$$
- Estimate of future cumulative claims:
$$\hat{C}_{i,J} = C_{i,J-i} \cdot \hat{f}_{J-i} \cdot \cdots \cdot \hat{f}_{J-1}, 0 \leq i \leq I$$

Chain ladder estimate

- Estimate of development factors:
$$\hat{f}_j = \frac{\sum_{i=0}^{I-j-1} C_{i,j+1}}{\sum_{k=0}^{I-j-1} C_{k,j}} = \sum_{i=0}^{I-j-1} \frac{C_{i,j}}{\sum_{k=0}^{I-j-1} C_{k,j}} \cdot \frac{C_{i,j+1}}{C_{i,j}} = \sum_{i=0}^{I-j-1} \frac{C_{i,j}}{\sum_{k=0}^{I-j-1} C_{k,j}} \cdot F_{i,j}$$
- Estimator $\hat{f}_j$ is an unbiased D_{j+1}-measurable estimator for f_j, which has a minimal conditional variance among all unbiased linear combinations of $F_{i,j}$, $0 \leq i \leq I - j - 1$, conditional on D_j (for proof see [2])

Chain ladder - LoB 1 (values in millions)

Paid triangle	0	1	2	3	4	5	6	7	8	9	10	11
1994	22,47	35,69	40,32	43,05	44,63	45,86	46,81	47,60	48,18	48,68	49,17	49,54
1995	27,13	42,80	47,99	50,59	52,38	53,68	54,71	55,61	56,33	56,97	57,53	
1996	30,91	47,93	53,66	56,75	58,67	60,07	61,11	62,02	62,70	63,25		
1997	40,82	64,97	73,82	78,39	81,30	83,36	85,04	86,34	87,39			
1998	51,96	82,05	92,69	98,40	102,13	104,77	106,75	108,30				
1999	60,46	97,77	111,34	118,59	122,99	126,12	128,54					
2000	78,81	127,03	144,80	153,79	159,61	163,48						
2001	73,39	120,04	137,91	147,42	153,55							
2002	91,41	147,71	168,73	179,36								
2003	95,98	155,16	177,97									
2004	79,40	126,99										
2005	90,35											

Ratios	0	1	2	3	4	5	6	7	8	9	10
1994	1,589	1,130	1,068	1,037	1,027	1,021	1,017	1,012	1,011	1,010	1,008
1995	1,578	1,121	1,054	1,035	1,025	1,019	1,016	1,013	1,011	1,010	
1996	1,551	1,119	1,058	1,034	1,024	1,017	1,015	1,011	1,009		
1997	1,592	1,136	1,062	1,037	1,025	1,020	1,015	1,012			
1998	1,579	1,130	1,062	1,038	1,026	1,019	1,014				
1999	1,617	1,139	1,065	1,037	1,025	1,019					
2000	1,612	1,140	1,062	1,038	1,024						
2001	1,636	1,149	1,069	1,042							
2002	1,616	1,142	1,063								
2003	1,617	1,147									
2004	1,599										

	0	1	2	3	4	5	6	7	8	9	10
Dev. factors	1,606	1,139	1,063	1,038	1,025	1,019	1,015	1,012	1,010	1,010	1,008

Chain ladder - LoB 2 (values in millions)

Paid triangle	0	1	2	3	4	5	6	7	8	9	10	11
1994	32,82	49,12	54,11	56,81	58,10	59,03	59,65	59,99	60,34	60,74	60,92	60,83
1995	46,68	73,66	85,68	90,32	92,92	94,35	95,19	95,68	96,08	96,24	96,39	
1996	50,03	73,84	82,64	85,94	87,09	87,82	88,27	88,59	88,77	88,93		
1997	46,22	71,82	82,14	86,53	88,55	89,65	90,36	90,78	91,14			
1998	58,59	92,75	107,53	113,06	115,59	117,24	118,38	119,06				
1999	131,86	205,89	235,35	246,98	252,71	255,78	257,60					
2000	207,47	325,96	369,81	389,08	398,82	403,95						
2001	202,25	319,67	362,66	378,73	386,67							
2002	496,69	802,24	918,69	966,00								
2003	561,90	894,08	1 021,80									
2004	873,37	1 399,49										
2005	663,10											

Ratios	0	1	2	3	4	5	6	7	8	9	10
1994	1,496	1,102	1,050	1,023	1,016	1,011	1,006	1,006	1,007	1,003	0,999
1995	1,578	1,163	1,054	1,029	1,015	1,009	1,005	1,004	1,002	1,002	
1996	1,476	1,119	1,040	1,013	1,008	1,005	1,004	1,002	1,002		
1997	1,554	1,144	1,054	1,023	1,012	1,008	1,005	1,004			
1998	1,583	1,159	1,051	1,022	1,014	1,010	1,006				
1999	1,561	1,143	1,049	1,023	1,012	1,007					
2000	1,571	1,135	1,052	1,025	1,013						
2001	1,581	1,134	1,044	1,021							
2002	1,615	1,145	1,052								
2003	1,591	1,143									
2004	1,602										

	0	1	2	3	4	5	6	7	8	9	10
Dev. factors	1,591	1,141	1,050	1,023	1,013	1,008	1,005	1,004	1,003	1,002	0,999

Chain ladder - LoB 3 (values in millions)

Paid triangle	0	1	2	3	4	5	6	7	8	9	10	11
1994	93,10	147,64	163,50	169,91	173,70	176,08	177,85	179,03	179,93	180,80	181,39	181,90
1995	91,04	144,65	159,42	165,93	169,67	171,95	173,64	174,93	175,63	176,20	176,87	
1996	86,50	143,91	163,08	171,63	176,86	179,68	181,75	183,44	184,57	185,54		
1997	100,37	171,55	194,78	204,42	209,49	213,08	215,77	217,42	218,87			
1998	103,77	170,38	193,09	202,21	206,69	209,40	211,50	213,10				
1999	109,12	178,87	202,43	212,72	218,15	221,60	223,64					
2000	111,02	178,77	201,34	211,09	216,20	219,20						
2001	123,10	212,31	244,16	257,30	264,17							
2002	140,16	234,07	266,87	280,85								
2003	156,45	259,36	293,88									
2004	152,88	256,85										
2005	176,47											

Ratios	0	1	2	3	4	5	6	7	8	9	10
1994	1,586	1,107	1,039	1,022	1,014	1,010	1,007	1,005	1,005	1,003	1,003
1995	1,589	1,102	1,041	1,023	1,013	1,010	1,007	1,004	1,003	1,004	
1996	1,664	1,133	1,052	1,030	1,016	1,012	1,009	1,006	1,005		
1997	1,709	1,135	1,050	1,025	1,017	1,013	1,008	1,007			
1998	1,642	1,133	1,047	1,022	1,013	1,010	1,008				
1999	1,639	1,132	1,051	1,026	1,016	1,009					
2000	1,610	1,126	1,048	1,024	1,014						
2001	1,725	1,150	1,054	1,027							
2002	1,670	1,140	1,052								
2003	1,658	1,133									
2004	1,680										

	0	1	2	3	4	5	6	7	8	9	10
Dev. factors	1,655	1,131	1,049	1,025	1,015	1,011	1,008	1,006	1,004	1,004	1,003

Chain ladder - LoB 4 (values in millions)

Paid triangle	0	1	2	3	4	5	6	7	8	9	10	11
1994	95,52	155,29	177,79	190,09	197,85	203,59	208,08	211,76	214,62	217,02	219,19	220,87
1995	99,88	166,86	193,42	208,48	218,63	226,05	232,06	237,01	241,16	244,84	248,12	
1996	95,88	159,31	184,46	198,14	206,77	213,09	218,07	222,01	225,17	227,89		
1997	96,37	162,52	188,20	202,85	212,41	219,48	225,02	229,16	232,47			
1998	94,65	160,03	186,78	201,32	210,85	217,50	222,66	226,63				
1999	98,66	168,09	196,08	211,56	221,66	228,84	234,41					
2000	98,81	169,30	197,92	213,90	224,21	231,70						
2001	101,45	175,88	207,93	225,44	236,69							
2002	106,69	182,13	212,89	229,64								
2003	109,73	194,14	230,92									
2004	115,17	200,33										
2005	120,03											

Ratios	0	1	2	3	4	5	6	7	8	9	10
1994	1,626	1,145	1,069	1,041	1,029	1,022	1,018	1,014	1,011	1,010	1,008
1995	1,671	1,159	1,078	1,049	1,034	1,027	1,021	1,017	1,015	1,013	
1996	1,661	1,158	1,074	1,044	1,031	1,023	1,018	1,014	1,012		
1997	1,686	1,158	1,078	1,047	1,033	1,025	1,018	1,014			
1998	1,691	1,167	1,078	1,047	1,032	1,024	1,018				
1999	1,704	1,167	1,079	1,048	1,032	1,024					
2000	1,713	1,169	1,081	1,048	1,033						
2001	1,734	1,182	1,084	1,050							
2002	1,707	1,169	1,079								
2003	1,769	1,189									
2004	1,739										

	0	1	2	3	4	5	6	7	8	9	10
Dev. factors	1,702	1,167	1,078	1,047	1,032	1,024	1,019	1,015	1,013	1,012	1,008

Results of chain ladder

- Triangles are stable
- Mack chain ladder estimated the reserve quite well (we do not consider tail development)

[Millions]	Chain ladder reserve	True reserve	Difference	Difference [%]	$\sqrt{MSEP}$
LoB 1	268,76	259,77	8,99	3,5%	3,79
LoB 2	1 240,49	1 310,85	-70,36	-5,4%	25,15
LoB 3	361,04	380,31	-19,27	-5,1%	11,07
LoB 4	478,89	485,71	-6,82	-1,4%	11,21
Total	**2 349,18**	**2 436,64**	**-87,46**	**-3,6%**	**-**

4. Presentation of a simple neural network model based on chain ladder approach

Neural network applied to chain ladder reserving

- Proposed model can be found in [3]
- Available features of claims $\boldsymbol{x} \in \boldsymbol{X}$ determined by LoB, cc, AQ, age and inj_part
- $C_{i,j}(\boldsymbol{x})$ is sum of all cumulative claims from accident year i within the first j development years and having feature value $\boldsymbol{x}$
- Basic idea is that claims with the same features $\boldsymbol{x}$ follow a pattern

$$C_{i,j}(\boldsymbol{x}) = f_{j-1}(\boldsymbol{x}) \cdot C_{i,j-1}(\boldsymbol{x}), \text{ if } C_{i,j-1}(\boldsymbol{x}) > 0$$

- Without considering different development factors depending on features $\boldsymbol{x}$, this would be the same as chain ladder

Model assumptions

- We assume independence of $\{C_{i,j}(\boldsymbol{x}), 0 \leq j \leq J, \boldsymbol{x} \in \boldsymbol{X}\}$ and $\{C_{k,j}(\boldsymbol{x}), 0 \leq j \leq J, \boldsymbol{x} \in \boldsymbol{X}\}$ for any different accident years $i \neq k$
- There exist positive parameters $f_0(\boldsymbol{x}), \ldots, f_{J-1}(\boldsymbol{x})$ such that for all $0 \leq i \leq I$ and $1 \leq j \leq J$ and $\boldsymbol{x} \in \boldsymbol{X}$
$$\mathrm{E}\left(C_{i,j}(\boldsymbol{x})|D_{j-1}\right) = f_{j-1}(\boldsymbol{x}) \cdot C_{i,j-1}(\boldsymbol{x}) + \mathrm{E}\left(C_{i,j}(\boldsymbol{x})|D_{j-1}\right) \cdot \mathbf{1}_{\{C_{i,j-1}(\boldsymbol{x})=\mathbf{0}\}}$$
- There exist positive parameters $\sigma_0^2, \ldots, \sigma_{J-1}^2$ such that for all $0 \leq i \leq I$ and $1 \leq j \leq J$ and $\boldsymbol{x} \in \boldsymbol{X}$
$$\mathrm{Var}\left(C_{i,j}(\boldsymbol{x})|D_{j-1}\right) = \sigma_{j-1}^2 \cdot C_{i,j-1}(\boldsymbol{x}) + \mathrm{Var}\left(C_{i,j}(\boldsymbol{x})|D_{j-1}\right) \cdot \mathbf{1}_{\{C_{i,j-1}(\boldsymbol{x})=\mathbf{0}\}}$$

Minimization problem

- Based on the assumptions, we want to estimate parameters for j-th development year, which minimize the following loss function:

$$L_j = \sum_{i=0}^{I-j} \sum_{\boldsymbol{x}:\, C_{i,j-1}(\boldsymbol{x})>0} \frac{\left[C_{i,j}(\boldsymbol{x}) - f_{j-1}(\boldsymbol{x}) \cdot C_{i,j-1}(\boldsymbol{x})\right]^2}{\sigma_{j-1}^2 \cdot C_{i,j-1}(\boldsymbol{x})} = \frac{1}{\sigma_{j-1}^2} \sum_{i=0}^{I-j} \sum_{\boldsymbol{x}:\, C_{i,j-1}(\boldsymbol{x})>0} C_{i,j-1}(\boldsymbol{x}) \cdot \left[\frac{C_{i,j}(\boldsymbol{x})}{C_{i,j-1}(\boldsymbol{x})} - f_{j-1}(\boldsymbol{x})\right]^{\mathbf{2}}$$

- σ_{j-1}^2 is a parameter, which we do not model, and it is assumed to not depend on $\boldsymbol{x}$, therefore this parameter is not considered in the minimization

- Loss function would be weighted mean squares, which is inconvenient for built-in functions, where we can find for example an unweighted mean square loss function, so we rewrite the loss function to the form

$$\sigma_{j-1}^2 \cdot L_j = \sum_{i=0}^{I-j} \sum_{\boldsymbol{x}:\, C_{i,j-1}(\boldsymbol{x})>0} \left[\frac{C_{i,j}(\boldsymbol{x})}{\sqrt{C_{i,j-1}(\boldsymbol{x})}} - f_{j-1}(\boldsymbol{x}) \cdot \sqrt{C_{i,j-1}(\boldsymbol{x})}\right]^{\mathbf{2}}$$

- Meaning: we will model ratio $\frac{C_{i,j}(\boldsymbol{x})}{\sqrt{C_{i,j-1}(\boldsymbol{x})}}$ with development factors having an offset $\sqrt{C_{i,j-1}(\boldsymbol{x})}$

Structure of neural network model

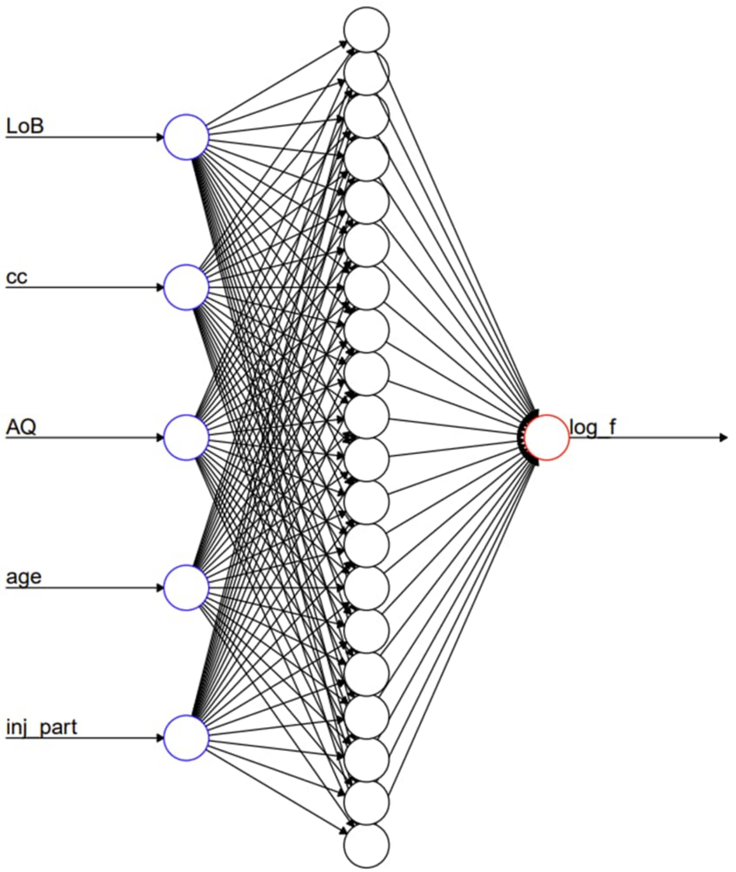
- To model logarithms of development factors depending on features $\boldsymbol{x}$ we can use a shallow neural network with one hidden layer and hyperbolic tangent activation function
- Hyperbolic tangent is a continuous function in range $(-1, 1)$, which together with weights provides reasonable modelling for the middle layer

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

- Exponential is suitable for modelling positive values of "development factors"
- These link functions lead us to formula

$$f_{j-1}(\boldsymbol{x}) \cdot \sqrt{C_{i,j-1}(\boldsymbol{x})} = \exp\left\{\beta_0 + \sum_{k=1}^{q} \beta_k \cdot \tanh\left(w_{k,0} + \sum_{m=1}^{d} w_{k,m} \cdot x_m\right)\right\}$$

where d is dimension of $\boldsymbol{X}$ and q is number of neurons



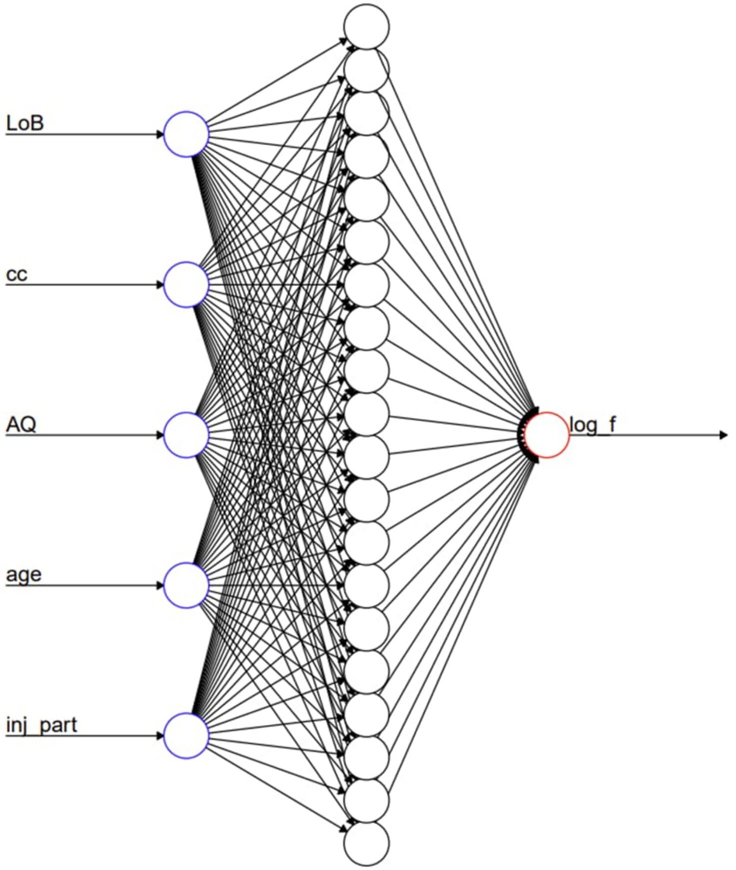
Number of hidden neurons

- A number of neurons needs to be chosen
- Too few neurons could fail to capture the underlying patterns
- Too many neurons could lead to an overfitting
- Consider the number of parameters in the model for **one development year**:

$$(d+1) \cdot q + q + 1$$

where $d = 106$ and for $q = 20$ we have 2 161 parameters

- With 11 development years we have overall 23 771 parameters (assuming we use the same number of hidden neurons)



Zero claims (IBNyR and zero IBNeR claims)

- The proposed model can estimate future development only for features $\boldsymbol{x}$, for which cumulative payments $C_{i,j-1}(\boldsymbol{x})$ are positive, therefore we need a way to estimate also the rest, where it is zero
- A simple approach with two steps can be used:
  1) Starting from the last diagonal, we estimate a proportion of zero claims turning into non-zero claims
  2) Then for the remaining development years we model these non-zero amounts like in chain ladder
- Future development of zero claims can then be estimated according to the two steps above as

$$\hat{C}^*_{i,J} = C_{i,I-i} \cdot \prod_{j=I-i}^{J-1} \hat{g}_j^{(i)}$$

- Formulas for $\hat{g}_j^{(i)}$ are described on following slides

Zero claims – step 1

- More precisely, we denote for $0 \leq k \leq i$ at development period $I - i$

$$X_k^{(i)} = \{\boldsymbol{x} \in \boldsymbol{X} : C_{k,I-i}(\boldsymbol{x}) = 0\},$$

$$C_{k,j}^{(*i)} = \sum_{\boldsymbol{x} \in X_k^{(i)}} C_{k,j}(\boldsymbol{x}) \text{ for } j > I - i$$

and the proportion for i-th accident year in 1) is estimated as

$$\hat{g}_{I-i}^{(i)} = \frac{\sum_{k=1}^{i-1} C_{k,I-i+1}^{(*i)}}{\sum_{k=1}^{i-1} C_{k,I-i}}$$

- This gives us the first factor for i-th accident year

Zero claims – step 2

- For following development years, we estimate factors similarly like in chain ladder:

$$\hat{g}_j^{(i)} = \frac{\sum_{k=1}^{i-1} C_{k,j+1}^{(*i)}}{\sum_{k=1}^{i-1} C_{k,j}^{(*i)}}$$

- An example of factors $\hat{g}_j$ can be seen in the table below for Lob 2 (orange cells are proportions from step 1)

- Example: Lob 2 AY 2005 (millions):

$663{,}10 \cdot 22{,}63\,\% \cdot 143{,}75\,\% \cdot \dots \cdot 99{,}75\,\%$

- We obtain result of 267,55 m

	0	1	2	3	4	5	6	7	8	9	10
1994											
1995											0,00%
1996										0,00%	100,00%
1997									0,01%	100,00%	100,00%
1998								0,04%	123,76%	145,21%	106,74%
1999							0,02%	249,96%	114,33%	123,41%	106,56%
2000						0,02%	181,26%	189,48%	110,29%	120,07%	104,86%
2001					0,07%	136,83%	124,55%	130,56%	105,46%	112,37%	97,74%
2002				0,08%	225,99%	126,32%	116,24%	139,07%	103,99%	112,91%	98,07%
2003			0,15%	211,36%	143,23%	119,35%	109,51%	117,56%	104,05%	108,72%	99,08%
2004		0,38%	164,81%	136,92%	118,28%	110,46%	105,15%	109,66%	107,29%	106,05%	99,55%
2005	22,63%	143,75%	110,77%	104,64%	102,44%	101,50%	100,82%	101,04%	100,71%	100,59%	99,75%

5. Application of the model in R

Neural networks in R

- Package keras3 acts as a bridge to Python, which needs to be installed with necessary backend libraries (TensorFlow or PyTorch)
- This allows us to use R syntax that mirrors Keras Python API
- Computations related to neural networks are run in Python and its backend
- Results are then available in R
- Important: versions of programs and packages must fulfill some requirements to be consistent with each other
- A complete code of the model can be found in [10]

Parameters in base scenario

- Packages: doParallel, reticulate, keras3
- Parameters that need to be set:
  - random seed,
  - number of neurons,
  - validation split,
  - batch size,
  - number of epochs.

```r
# Set random seed
RandomSeed <- 1992
set_random_seed(RandomSeed)

# Number of hidden neurons and other parameters
q <- 20
dropout_rate <- 0.5
val_split <- 0.1
batch <- 10000
epochs <- 110
```

Data for the model

- Features are stored in df.X
- Indexes of the data used for training / validation: *estimate*
- Indexes for prediction: *non_estimate*
- Responses are calculated as $C_j / \sqrt{C_{j-1}}$

```r
df.X <- as.matrix(df_encoded_NN[, !names(df_encoded_NN) %in%
                                c("AY", "C0", "C1", "C2", "C3", "C4", "C5", "C6", "C7", "C8", "C9", "C10", "C11", "RepDel", "age")])
for (j in 1 : max_dev_year) {
  # Filter values in the observed responses (non_estimate) and values for prediction (estimate)
  estimate <- df_encoded_NN[[paste0("C", j - 1)]] > 0 & df_encoded_NN$AY >= year_max - j + 1
  non_estimate <- df_encoded_NN[[paste0("C", j - 1)]] > 0 & df_encoded_NN$AY < year_max - j + 1

  # Calculate observed responses and volumes for the filtered data
  df.W_non_estimate <- as.matrix(sqrt(df_encoded_NN[[paste0("C", j - 1)]][non_estimate]))  # Volumes used as offsets for training
  df.W_estimate <- as.matrix(sqrt(df_encoded_NN[[paste0("C", j - 1)]][estimate])) # Volumes used as offsets for prediction
  df.Y <- as.matrix(df_encoded_NN[[paste0("C", j)]][non_estimate] / df.W_non_estimate)  # Observed responses
```

Definition of neural network

- Neural network can be defined as below using functions, such as layer_input, layer_dense, layer_dropout and layer_multiply
- Offsets are included as a layer, which is not trainable

```r
features <- layer_input(shape = c(ncol(df.x)))

net <- features %>%
  layer_dense(units = q, activation = 'tanh') %>%
  layer_dropout(rate = dropout_rate) %>%
  layer_dense(units = 1, activation = 'exponential')

volumes <- layer_input(shape = c(1))

offset <- volumes %>%
  layer_dense(units = 1, activation = 'linear', use_bias = FALSE, trainable = FALSE)

merged <- list(net, offset) %>%
  layer_multiply()

model <- keras_model(inputs = list(features, volumes), outputs = merged)
```

Important step

- Offsets form a non-trainable layer, but its weight is still randomly generated in the beginning, see the formula below and parameter k:

$$f_{j-1}(\boldsymbol{x}) \cdot \sqrt{C_{i,j-1}(\boldsymbol{x})} = \underline{k} \cdot \exp\left\{\beta_0 + \sum_{k=1}^{q} \beta_k \cdot \tanh\left(w_{k,0} + \sum_{m=1}^{d} w_{k,m} \cdot x_m\right)\right\}$$

- Parameter k must be set equal to 1, for the model to work as described earlier

```
# Change the last weight to 1
weights <- get_weights(model)
weights[[5]] <- as.matrix(1)
set_weights(model, weights)
```

Fitting model and prediction

- Because the output of the model is $f_{j-1}(\boldsymbol{x}) \cdot \sqrt{C_{i,j-1}(\boldsymbol{x})}$, multiplying it again by $\sqrt{C_{i,j-1}(\boldsymbol{x})}$ provides an estimate of $C_{i,j}(\boldsymbol{x})$
- Verbose = 0 means no printing of outputs during training of the neural network and prediction

```r
  model %>% compile(loss = 'mse', optimizer = 'rmsprop')

  # Fit the neural network
  fit <- model %>% fit(list(df.X[non_estimate, ], df.W_non_estimate),
                       df.Y,
                       epochs = epochs,
                       batch_size = batch,
                       validation_split = val_split,
                       verbose = 0)

  pred <- as.vector(model %>% predict(list(df.X[estimate, ], df.W_estimate), verbose = 0)) * df.W_estimate

# Create a logical index from the estimate vector and save estimated values
  true_indices <- which(estimate)
  df_encoded_NN[[paste0("C", j)]][true_indices] <- pred
}
```

Model summary (R output)

- Below is a summary generated in R to the described model

```
> summary(model)
Model: "functional"
```

Layer (type)	Output Shape	Param #	Connected to	Trainable
input_layer (InputLayer)	(None, 106)	0	-	-
dense (Dense)	(None, 20)	2,140	input_layer[0][0]	Y
dropout (Dropout)	(None, 20)	0	dense[0][0]	-
input_layer_1 (InputLayer)	(None, 1)	0	-	-
dense_1 (Dense)	(None, 1)	21	dropout[0][0]	Y
dense_2 (Dense)	(None, 1)	1	input_layer_1[0][0]	N
multiply (Multiply)	(None, 1)	0	dense_1[0][0], dense_2[0][0]	-

```
Total params: 2,162 (8.45 KB)
Trainable params: 2,161 (8.44 KB)
Non-trainable params: 1 (4.00 B)
```

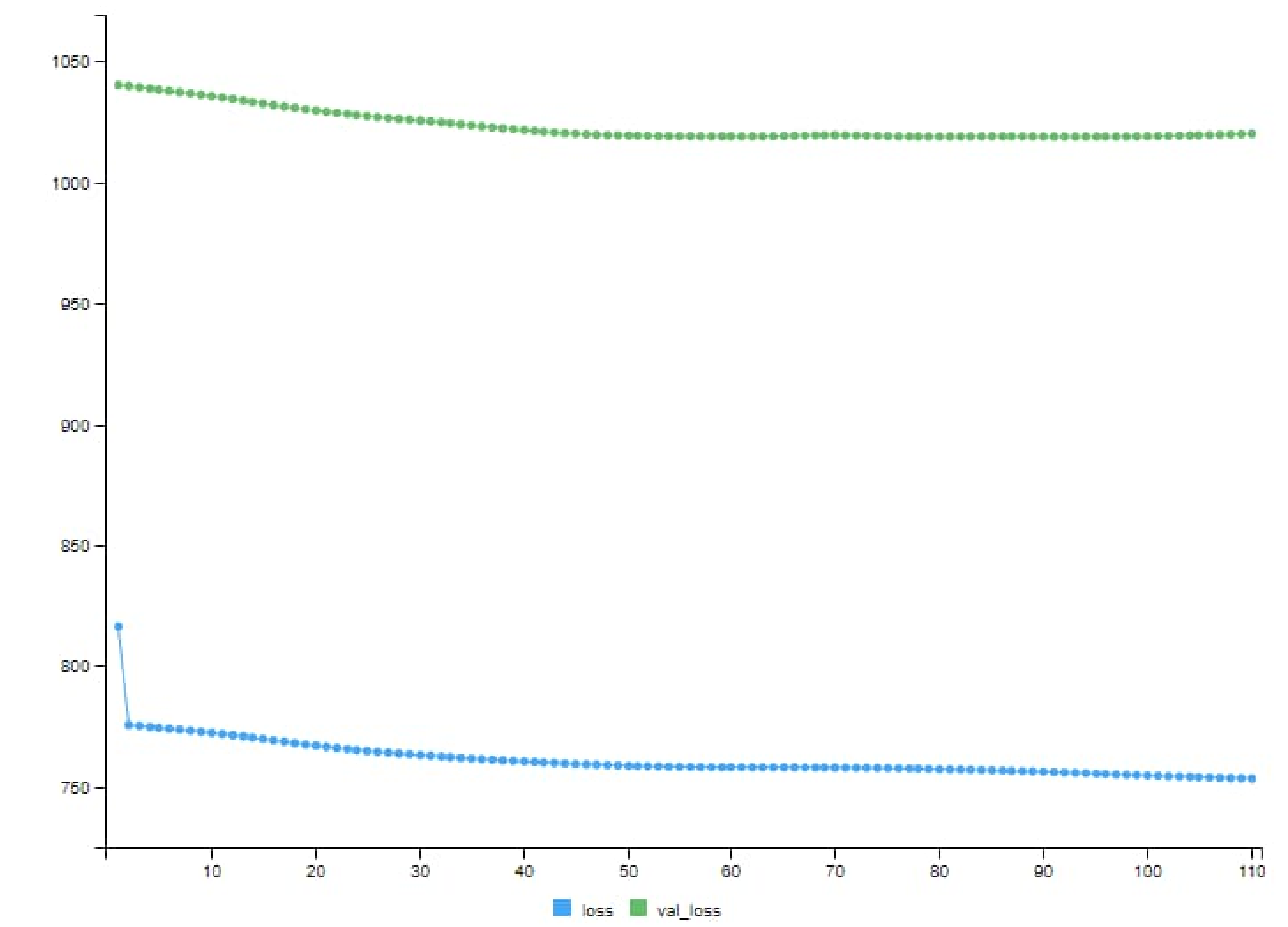
Other R outputs

Below is an output for the first development year (it appears when verbose is not set to zero)

```
Epoch 109/110

  1/226  31s 140ms/step - loss: 378.1052
 18/226  0s 3ms/step - loss: 640.6549
 32/226  0s 3ms/step - loss: 667.6657
 45/226  0s 4ms/step - loss: 718.8270
 60/226  0s 4ms/step - loss: 737.5414
 76/226  0s 3ms/step - loss: 740.6403
 89/226  0s 4ms/step - loss: 743.5276
102/226  0s 4ms/step - loss: 745.2264
117/226  0s 4ms/step - loss: 745.2510
134/226  0s 4ms/step - loss: 745.6368
148/226  0s 4ms/step - loss: 745.0671
162/226  0s 4ms/step - loss: 744.1882
178/226  0s 4ms/step - loss: 743.3379
194/226  0s 3ms/step - loss: 741.9296
208/226  0s 4ms/step - loss: 740.3707
225/226  0s 3ms/step - loss: 741.4543
226/226  1s 5ms/step - loss: 741.5604 - val_loss: 1020.0165
Epoch 110/110

  1/226  34s 154ms/step - loss: 377.5583
 15/226  0s 4ms/step - loss: 630.9404
 29/226  0s 4ms/step - loss: 656.8686
 45/226  0s 4ms/step - loss: 718.6977
 58/226  0s 4ms/step - loss: 736.2927
 73/226  0s 4ms/step - loss: 740.3409
 89/226  0s 4ms/step - loss: 743.3893
106/226  0s 3ms/step - loss: 745.2969
118/226  0s 4ms/step - loss: 745.1755
133/226  0s 4ms/step - loss: 745.5278
152/226  0s 3ms/step - loss: 744.7151
162/226  0s 4ms/step - loss: 744.0509
174/226  0s 4ms/step - loss: 743.3622
188/226  0s 4ms/step - loss: 742.4270
204/226  0s 4ms/step - loss: 740.6920
223/226  0s 4ms/step - loss: 741.2012
226/226  1s 5ms/step - loss: 741.4287 - val_loss: 1020.1652
>
```



6. Results - comparison to chain ladder result and true values

Results of the neural network model – zero claims

- The estimate for zero claims (IBNyR claims and zero IBNeR claims) can be seen below, there is only a small difference in absolute values
- For this reason, all following comparisons include also this estimate

[Millions]	Zero claims estimate	True reserve for zero claims	Difference	Difference [%]
LoB 1	38,45	35,20	3,25	9,2%
LoB 2	301,30	301,12	0,18	0,1%
LoB 3	90,07	92,52	-2,45	-2,6%
LoB 4	65,16	64,04	1,12	1,7%
Total	**494,98**	**492,88**	**2,10**	**0,4%**

Results of the neural network model (without dropout)

- Random seed 1992, 110 epochs, batch size 10 000
- This is our base scenario for a further testing
- A very similar result to the chain ladder estimate
- The table below includes zero claims from the previous slide

[Millions]	Neural network reserve	True reserve	Difference	Difference [%]
LoB 1	258,58	259,77	-1,19	-0,5%
LoB 2	1 307,57	1 310,85	-3,28	-0,3%
LoB 3	338,34	380,31	-41,97	-11,0%
LoB 4	456,84	485,71	-28,87	-5,9%
Total	**2 361,33**	**2 436,64**	**-75,31**	**-3,1%**

Detailed comparison (LoB 1)

- The neural network model seems to be more volatile in various accident years

[Millions]	True reserve	CL reserve	NN reserve	CL - True difference	NN - True difference	CL - True difference [%]	NN - True difference [%]
1994	0,00	0,00	0,00	0,00	0,00		
1995	0,52	0,44	-0,35	-0,08	-0,87	-15,4%	-167,3%
1996	0,87	1,10	0,29	0,23	-0,58	26,4%	-66,7%
1997	2,50	2,43	1,13	-0,07	-1,37	-2,8%	-54,8%
1998	3,75	4,35	2,86	0,60	-0,89	16,0%	-23,7%
1999	6,27	7,22	6,10	0,95	-0,17	15,2%	-2,7%
2000	12,06	12,50	9,86	0,44	-2,20	3,6%	-18,2%
2001	15,94	15,89	13,20	-0,05	-2,74	-0,3%	-17,2%
2002	24,35	26,06	20,84	1,71	-3,51	7,0%	-14,4%
2003	36,27	38,74	35,87	2,47	-0,40	6,8%	-1,1%
2004	44,33	49,15	53,23	4,82	8,90	10,9%	20,1%
2005	112,91	110,88	115,55	-2,03	2,64	-1,8%	2,3%
Total	**259,77**	**268,76**	**258,58**	**8,99**	**-1,19**	**3,5%**	**-0,5%**

Detailed comparison (LoB 2)

- The neural network model seems to be more volatile in various accident years

[Millions]	True reserve	CL reserve	NN reserve	CL - True difference	NN - True difference	CL - True difference [%]	NN - True difference [%]
1994	0,00	0,00	0,00	0,00	0,00		
1995	0,02	-0,14	-0,84	-0,16	-0,86	-800,0%	-4 300,0%
1996	0,19	0,06	-0,76	-0,13	-0,95	-68,4%	-500,0%
1997	0,82	0,33	-0,55	-0,49	-1,37	-59,8%	-167,1%
1998	1,37	0,89	-0,39	-0,48	-1,76	-35,0%	-128,5%
1999	4,25	3,22	1,53	-1,03	-2,72	-24,2%	-64,0%
2000	11,15	8,28	3,52	-2,87	-7,63	-25,7%	-68,4%
2001	12,16	13,00	5,75	0,84	-6,41	6,9%	-52,7%
2002	72,07	55,23	34,12	-16,84	-37,95	-23,4%	-52,7%
2003	121,27	112,39	105,29	-8,88	-15,98	-7,3%	-13,2%
2004	402,01	373,61	430,94	-28,40	28,93	-7,1%	7,2%
2005	685,54	673,62	728,96	-11,92	43,42	-1,7%	6,3%
Total	**1 310,85**	**1 240,49**	**1 307,57**	**-70,36**	**-3,28**	**-5,4%**	**-0,3%**

Detailed comparison (LoB 3)

- The neural network model seems to be more volatile in various accident years

[Millions]	True reserve	CL reserve	NN reserve	CL - True difference	NN - True difference	CL - True difference [%]	NN - True difference [%]
1994	0,00	0,00	0,00	0,00	0,00		
1995	0,32	0,49	-1,02	0,17	-1,34	53,1%	-418,8%
1996	1,42	1,17	-0,48	-0,25	-1,90	-17,6%	-133,8%
1997	2,84	2,37	-1,81	-0,47	-4,65	-16,5%	-163,7%
1998	4,00	3,50	-0,48	-0,50	-4,48	-12,5%	-112,0%
1999	6,36	5,42	2,19	-0,94	-4,17	-14,8%	-65,6%
2000	6,45	7,69	2,51	1,24	-3,94	19,2%	-61,1%
2001	15,65	13,30	7,13	-2,35	-8,52	-15,0%	-54,4%
2002	24,95	21,48	12,12	-3,47	-12,83	-13,9%	-51,4%
2003	34,79	37,94	32,74	3,15	-2,05	9,1%	-5,9%
2004	64,48	71,12	81,84	6,64	17,36	10,3%	26,9%
2005	219,05	196,56	203,60	-22,49	-15,45	-10,3%	-7,1%
Total	**380,31**	**361,04**	**338,34**	**-19,27**	**-41,97**	**-5,1%**	**-11,0%**

Detailed comparison (LoB 4)

- The neural network model seems to be more volatile in various accident years

[Millions]	True reserve	CL reserve	NN reserve	CL - True difference	NN - True difference	CL - True difference [%]	NN - True difference [%]
1994	0,00	0,00	0,00	0,00	0,00		
1995	2,65	1,90	-1,24	-0,75	-3,89	-28,3%	-146,8%
1996	4,10	4,46	2,65	0,36	-1,45	8,8%	-35,4%
1997	6,83	7,61	2,63	0,78	-4,20	11,4%	-61,5%
1998	10,57	10,92	6,62	0,35	-3,95	3,3%	-37,4%
1999	15,87	15,89	12,85	0,02	-3,02	0,1%	-19,0%
2000	19,64	21,71	16,68	2,07	-2,96	10,5%	-15,1%
2001	30,03	30,48	24,76	0,45	-5,27	1,5%	-17,5%
2002	40,09	41,71	33,71	1,62	-6,38	4,0%	-15,9%
2003	64,05	63,19	60,31	-0,86	-3,74	-1,3%	-5,8%
2004	99,05	97,43	104,92	-1,62	5,87	-1,6%	5,9%
2005	192,83	183,59	192,95	-9,24	0,12	-4,8%	0,1%
Total	**485,71**	**478,89**	**456,84**	**-6,82**	**-28,87**	**-1,4%**	**-5,9%**

Detailed comparison (all LoBs summarized)

- The neural network model seems to be more volatile in various accident years

[Millions]	True reserve	CL reserve	NN reserve	CL - True difference	NN - True difference	CL - True difference [%]	NN - True difference [%]
1994	0,00	0,00	0,00	0,00	0,00		
1995	3,52	2,69	-3,45	-0,83	-6,97	-23,6%	-198,0%
1996	6,59	6,79	1,70	0,20	-4,89	3,0%	-74,2%
1997	12,99	12,74	1,40	-0,25	-11,59	-1,9%	-89,2%
1998	19,69	19,66	8,60	-0,03	-11,09	-0,2%	-56,3%
1999	32,75	31,75	22,67	-1,00	-10,08	-3,1%	-30,8%
2000	49,31	50,18	32,57	0,87	-16,74	1,8%	-33,9%
2001	73,77	72,67	50,84	-1,10	-22,93	-1,5%	-31,1%
2002	161,46	144,48	100,79	-16,98	-60,67	-10,5%	-37,6%
2003	256,38	252,26	234,21	-4,12	-22,17	-1,6%	-8,6%
2004	609,88	591,30	670,93	-18,58	61,05	-3,0%	10,0%
2005	1 210,33	1 164,66	1 241,05	-45,67	30,72	-3,8%	2,5%
Total	**2 436,67**	**2 349,18**	**2 361,31**	**-87,49**	**-75,36**	**-3,6%**	**-3,1%**

Comparison of development factors

- Chain ladder estimates the future development pattern better

Dev factors LoB 1	0	1	2	3	4	5	6	7	8	9	10
True	1,626	1,139	1,064	1,038	1,025	1,018	1,014	1,011	1,009	1,008	1,007
Chain ladder	1,606	1,139	1,063	1,038	1,025	1,019	1,015	1,012	1,010	1,010	1,008
NN	1,574	1,191	1,078	1,028	1,023	1,011	1,020	1,015	1,008	1,010	0,995

Dev factors LoB 2	0	1	2	3	4	5	6	7	8	9	10
True	1,597	1,146	1,050	1,024	1,013	1,009	1,007	1,005	1,003	1,003	1,001
Chain ladder	1,591	1,141	1,050	1,023	1,013	1,008	1,005	1,004	1,003	1,002	0,999
NN	1,558	1,195	1,066	1,016	1,010	1,003	1,009	1,006	1,000	1,001	0,992

Dev factors LoB 3	0	1	2	3	4	5	6	7	8	9	10
True	1,709	1,135	1,051	1,025	1,015	1,010	1,008	1,006	1,005	1,004	1,002
Chain ladder	1,655	1,131	1,049	1,025	1,015	1,011	1,008	1,006	1,004	1,004	1,003
NN	1,628	1,185	1,066	1,016	1,014	1,004	1,010	1,007	0,997	1,001	0,995

Dev factors LoB 4	0	1	2	3	4	5	6	7	8	9	10
True	1,744	1,181	1,083	1,049	1,033	1,023	1,018	1,014	1,011	1,009	1,008
Chain ladder	1,702	1,167	1,078	1,047	1,032	1,024	1,019	1,015	1,013	1,012	1,008
NN	1,672	1,216	1,097	1,042	1,029	1,016	1,024	1,018	1,000	1,017	0,996

What other aspects should we consider?

- Performance of the neural network model may be affected by several factors
- It could be worth inspecting the following:
  - Different artificial data ✖
  - Real data ✖
  - Sensitivity to different random seed ✓
  - Sensitivity to different choices of hyperparameters (batch size, number of epochs) ✓
  - Regularization techniques ✓

✖ *Not considered in this presentation*

✓ *Considered in this presentation*

7. Stability considerations

Sensitivity to random seed

- Sensitivity to random seed was quite obvious during first attempts to calibrate the neural network model
- Previously presented results by LoBs were obtained with 20 neurons and random seed 1992

[Millions]	Reserve	Difference	Difference [%]
True value	2 436,66	-	-
20 neurons, random seed 1992	2 361,31	-75,35	-3,1%
20 neurons, random seed 1993	2 355,20	-81,46	-3,3%
20 neurons, random seed 1994	2 236,72	-199,94	-8,2%
20 neurons, random seed 1995	2 028,26	-408,40	-16,8%
20 neurons, random seed 1996	2 526,90	90,24	3,7%
20 neurons, random seed 1997	2 405,98	-30,68	-1,3%
20 neurons, random seed 1998	2 367,91	-68,75	-2,8%
20 neurons, random seed 1999	2 429,20	-7,46	-0,3%
20 neurons, random seed 2000	1 948,59	-488,07	-20,0%
20 neurons, random seed 2001	2 503,23	66,57	2,7%

- Unfortunately, we observe a high volatility in results only due to different choices of random seed

[Millions]	Difference	Difference [%]
Minimum	-488,07	-20,0%
Maximum	90,24	3,7%
Spread	578,31	23,7%
Average	-120,33	-4,9%
Standard deviation	181,88	7,7%
Number of negative differences	8	-

Sensitivity to number of neurons (1/2)

- How would the result look with a different number of hidden neurons? Let's see a case of 15 neurons instead of 20
- Because of a high sensitivity to the random seed, this scenario is also run with 10 random seeds

[Millions]	Reserve	Difference	Difference [%]
True value	2 436,66	-	-
15 neurons, random seed 1992	2 449,52	12,86	0,5%
15 neurons, random seed 1993	2 346,74	-89,92	-3,7%
15 neurons, random seed 1994	2 352,38	-84,28	-3,5%
15 neurons, random seed 1995	1 981,59	-455,07	-18,7%
15 neurons, random seed 1996	2 528,09	91,43	3,8%
15 neurons, random seed 1997	2 429,26	-7,40	-0,3%
15 neurons, random seed 1998	2 423,50	-13,16	-0,5%
15 neurons, random seed 1999	2 402,10	-34,56	-1,4%
15 neurons, random seed 2000	1 958,82	-477,84	-19,6%
15 neurons, random seed 2001	2 436,16	-0,50	0,0%

- A very similar result to the setting of 20 neurons
- Noticeably similar seeds 1995 and 2000

[Millions]	Difference	Difference [%]
Minimum	-477,84	-19,6%
Maximum	91,43	3,8%
Spread	569,27	23,4%
Average	-105,84	-4,3%
Standard deviation	186,69	7,6%
Number of negative differences	8	-

Sensitivity to number of neurons (2/2)

- Here is presented case of 25 neurons instead of 20
- Because of a high sensitivity to the random seed, this scenario is also run with 10 random seeds

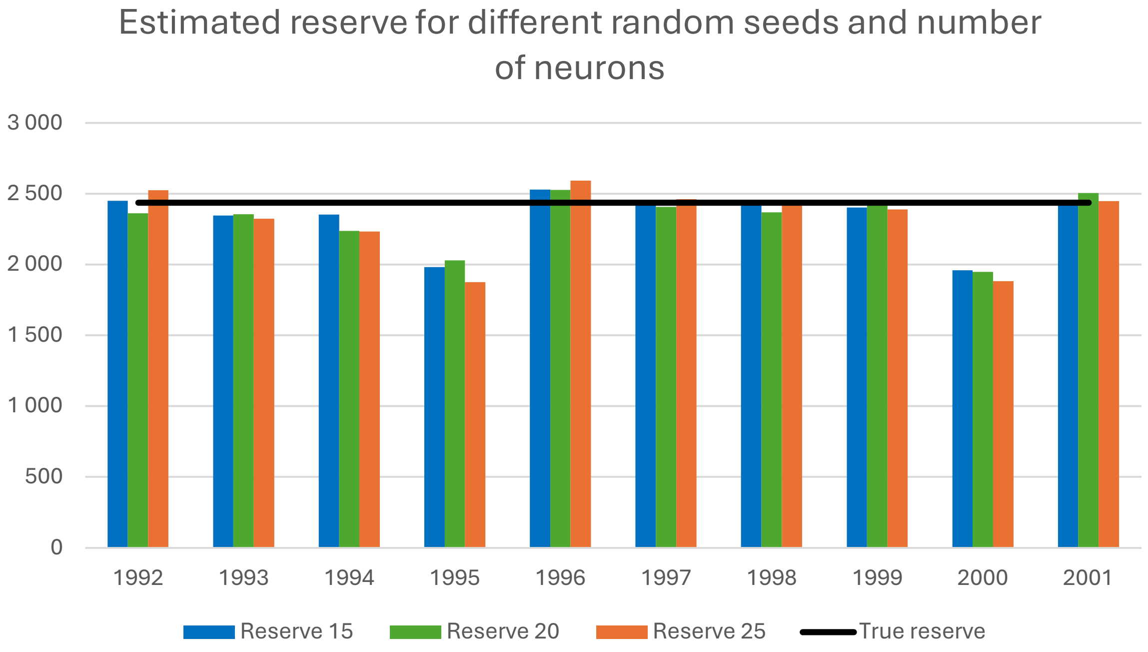
[Millions]	Reserve	Difference	Difference [%]
True value	2 436,66	-	-
25 neurons, random seed 1992	2 524,12	87,46	3,6%
25 neurons, random seed 1993	2 324,24	-112,42	-4,6%
25 neurons, random seed 1994	2 232,62	-204,04	-8,4%
25 neurons, random seed 1995	1 875,50	-561,16	-23,0%
25 neurons, random seed 1996	2 592,50	155,84	6,4%
25 neurons, random seed 1997	2 460,81	24,15	1,0%
25 neurons, random seed 1998	2 434,27	-2,39	-0,1%
25 neurons, random seed 1999	2 388,11	-48,55	-2,0%
25 neurons, random seed 2000	1 882,74	-553,92	-22,7%
25 neurons, random seed 2001	2 447,62	10,96	0,4%

- More volatile estimates to the setting of 20 neurons
- Seeds 1995 and 2000 are a bit worse

[Millions]	Difference	Difference [%]
Minimum	-561,16	-23,0%
Maximum	155,84	6,4%
Spread	717,00	29,4%
Average	-120,41	-4,9%
Standard deviation	237,91	9,4%
Number of negative differences	6	-

Sensitivity to number of neurons – overview of differences

- The model seems quite stable in terms of the number of hidden neurons
- In the case of 25 neurons, we observe a larger volatility in differences



Differences			
[Millions]	**15 neurons**	**20 neurons**	**25 neurons**
Random seed 1992	12,86	-75,35	87,46
Random seed 1993	-89,92	-81,46	-112,42
Random seed 1994	-84,28	-199,94	-204,04
Random seed 1995	-455,07	-408,40	-561,16
Random seed 1996	91,43	90,24	155,84
Random seed 1997	-7,40	-30,68	24,15
Random seed 1998	-13,16	-68,75	-2,39
Random seed 1999	-34,56	-7,46	-48,55
Random seed 2000	-477,84	-488,07	-553,92
Random seed 2001	-0,50	66,57	10,96

Different batch size and number of epochs

- Here is presented the case of 15 neurons and random seed 1992, but with different batch sizes instead of 10 000 and different numbers of epochs instead of 110
- More specifically, the left table keeps the number of epochs 110 and the right table keeps the batch size 10 000
- Batch size seems to be a significant factor, while for different numbers of epochs results are stable

Different batch sizes

[Millions]	Reserve	Difference	Difference [%]
True value	2 436,66	-	-
Batch 5 000	2 562,80	126,14	5,2%
Batch 7 500	2 507,62	70,96	2,9%
Batch 10 000	2 449,52	12,86	0,5%
Batch 12 500	2 519,39	82,73	3,4%
Batch 15 000	2 660,45	223,79	9,2%

Different numbers of epochs

[Millions]	Reserve	Difference	Difference [%]
True value	2 436,66	-	-
Epochs 90	2 467,51	30,85	1,3%
Epochs 100	2 460,27	23,61	1,0%
Epochs 110	2 449,52	12,86	0,5%
Epochs 120	2 449,82	13,16	0,5%
Epochs 130	2 450,56	13,90	0,6%

Application of a dropout layer

- Applied dropout is 50 %, other settings are the same as in the base scenario
- Dropout layer has improved the estimate significantly

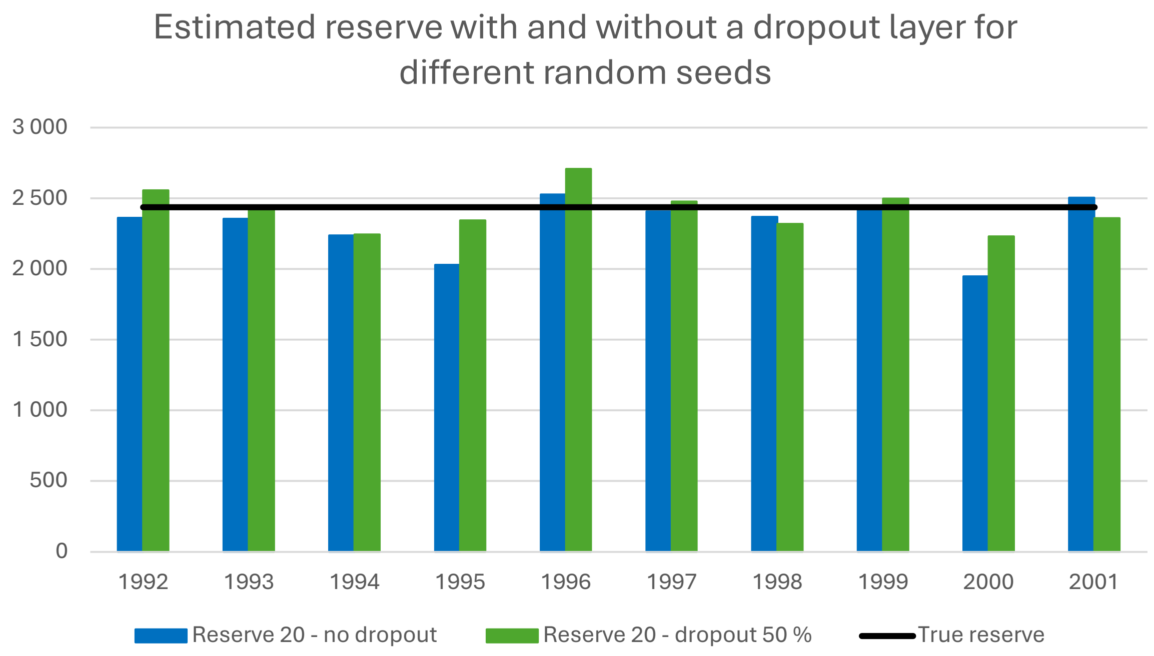
[Millions]	Reserve	Difference	Difference [%]
True value	2 436,66	-	-
20 neurons, random seed 1992	2 556,02	119,36	4,9%
20 neurons, random seed 1993	2 431,16	-5,50	-0,2%
20 neurons, random seed 1994	2 244,85	-191,81	-7,9%
20 neurons, random seed 1995	2 342,98	-93,68	-3,8%
20 neurons, random seed 1996	2 708,08	271,42	11,1%
20 neurons, random seed 1997	2 476,47	-79,55	-3,1%
20 neurons, random seed 1998	2 318,44	-112,72	-4,6%
20 neurons, random seed 1999	2 497,22	252,37	11,2%
20 neurons, random seed 2000	2 229,25	-207,41	-8,5%
20 neurons, random seed 2001	2 359,14	-196,88	-8,1%

- However, the volatility is still too high for different random seeds

[Millions]	Difference	Difference [%]
Minimum	-207,41	-8,5%
Maximum	271,42	11,1%
Spread	478,83	19,7%
Average	-24,44	-1,0%
Standard deviation	170,89	6,7%
Number of negative differences	7	-

Application of a dropout layer

- Below is a comparison for the case of 20 neurons with and without a dropout layer
- The largest improvements can be seen for the "worst" seeds: 1995 and 2000

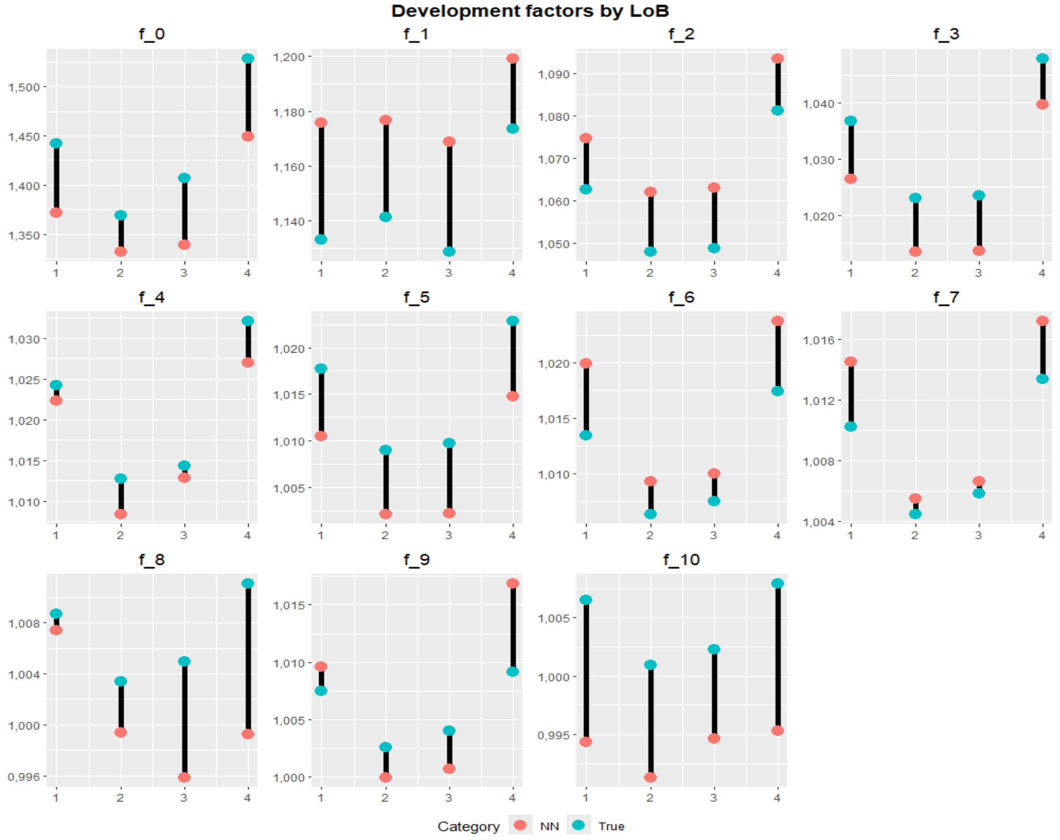


Differences		
[Millions]	**Reserve 20 - no dropout**	**Reserve 20 – 50% dropout**
Random seed 1992	-75,35	119,36
Random seed 1993	-81,46	-5,50
Random seed 1994	-199,94	-191,81
Random seed 1995	-408,40	-93,68
Random seed 1996	90,24	271,42
Random seed 1997	-30,68	-79,55
Random seed 1998	-68,75	-112,72
Random seed 1999	-7,46	252,37
Random seed 2000	-488,07	-207,41
Random seed 2001	66,57	-196,88

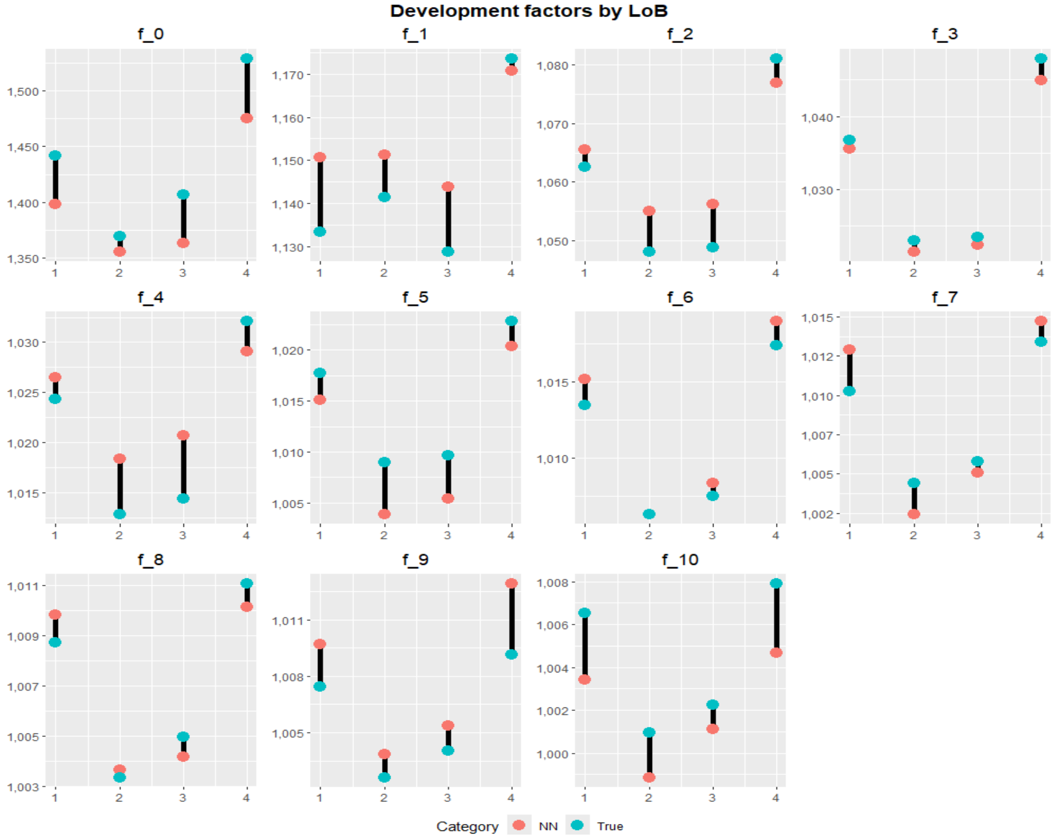
Comparison of development factors (LoB)

- Only non-zero claims are considered
- Some improvements thanks to the dropout layer can be observed

No dropout



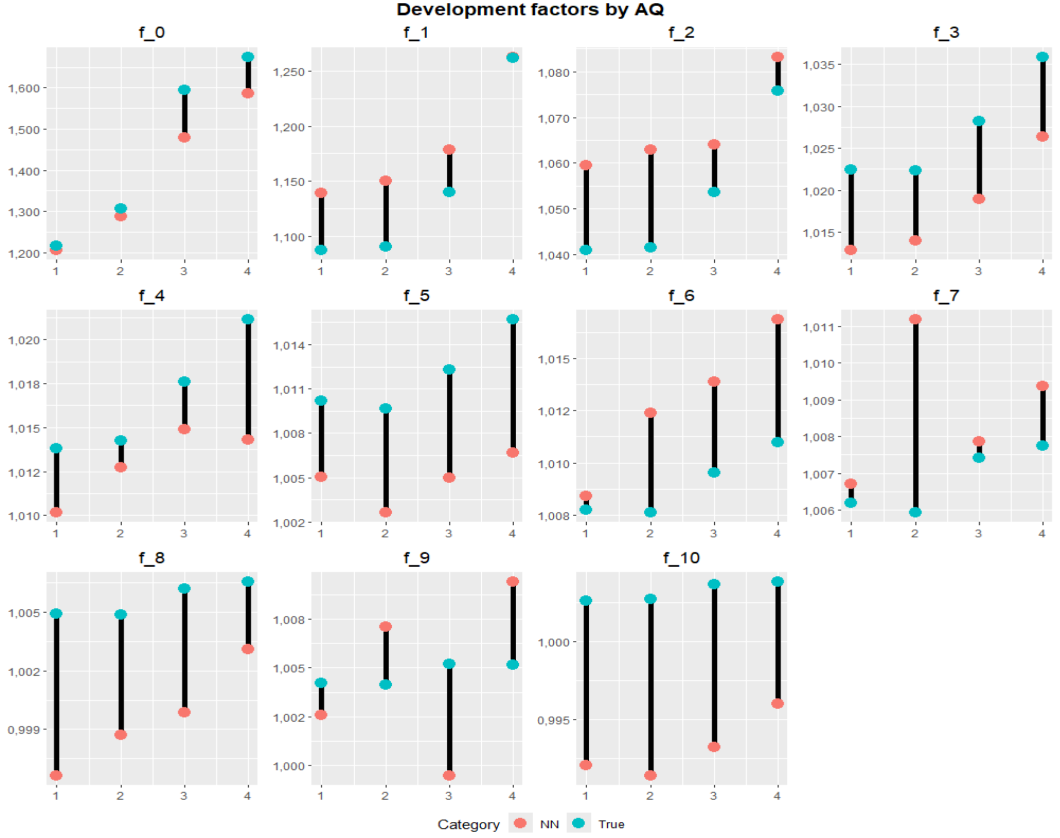
Dropout



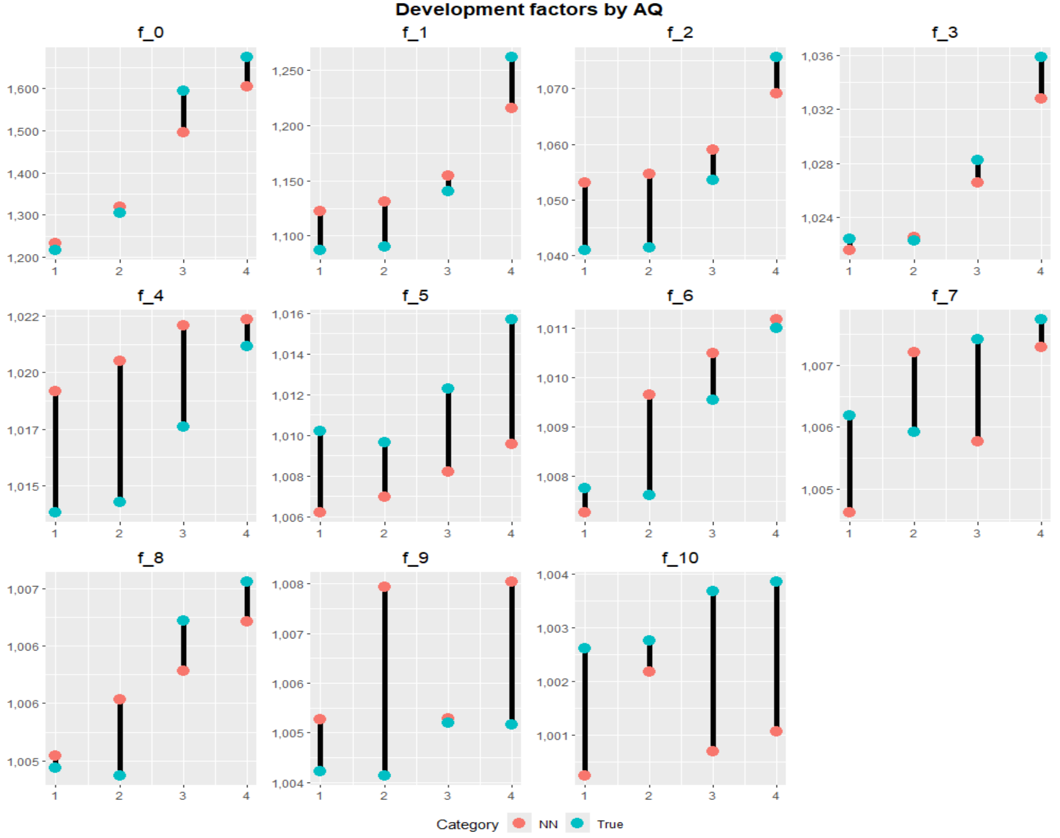
Comparison of development factors (AQ)

- Some improvements thanks to the dropout layer can be observed

No dropout



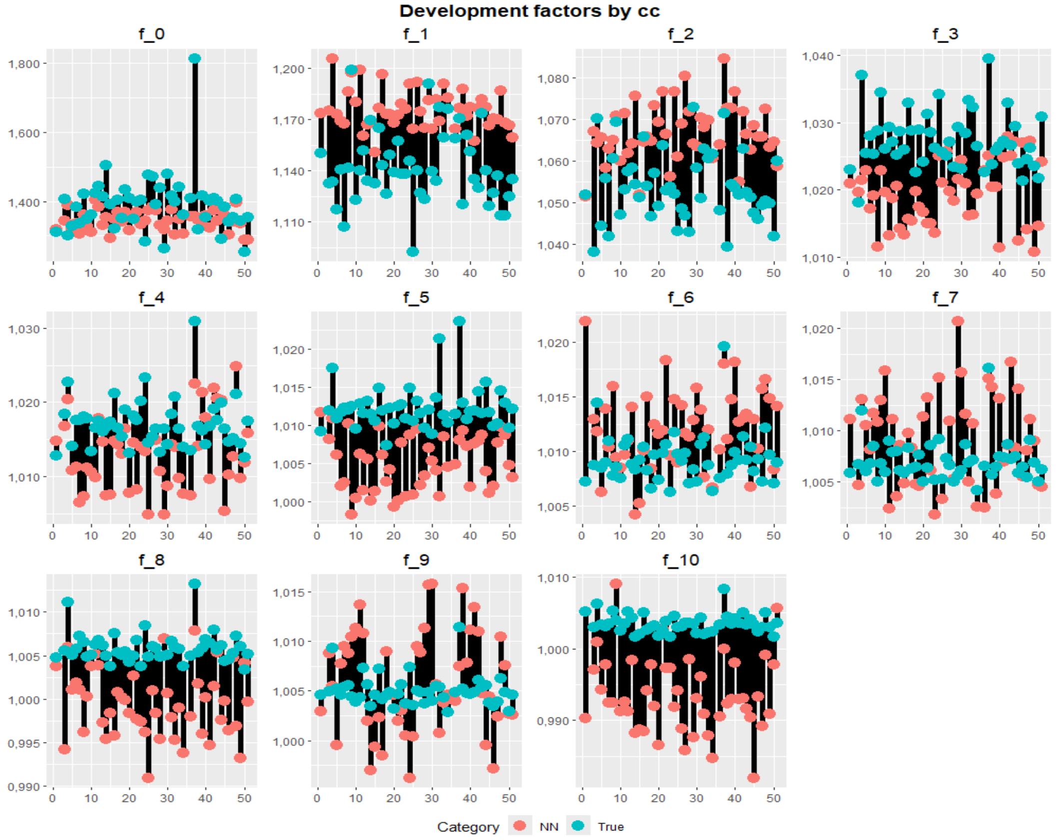
Dropout



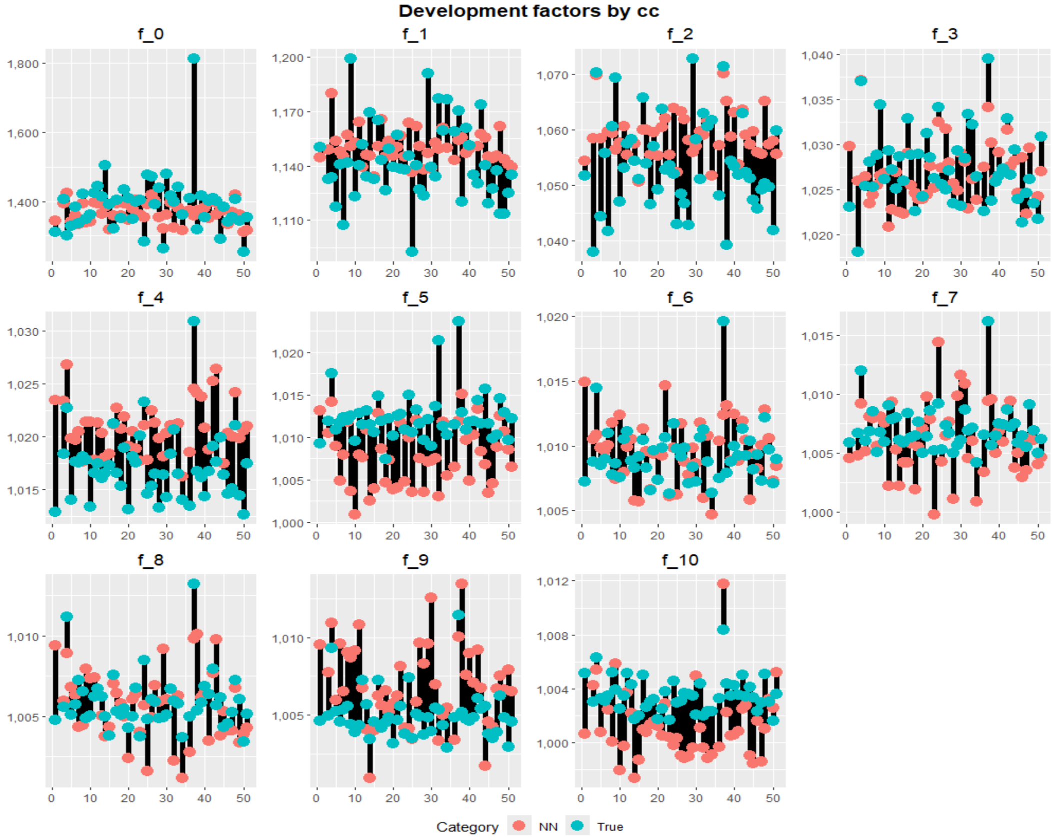
Comparison of development factors (cc)

- Some improvements thanks to the dropout layer can be observed

No dropout



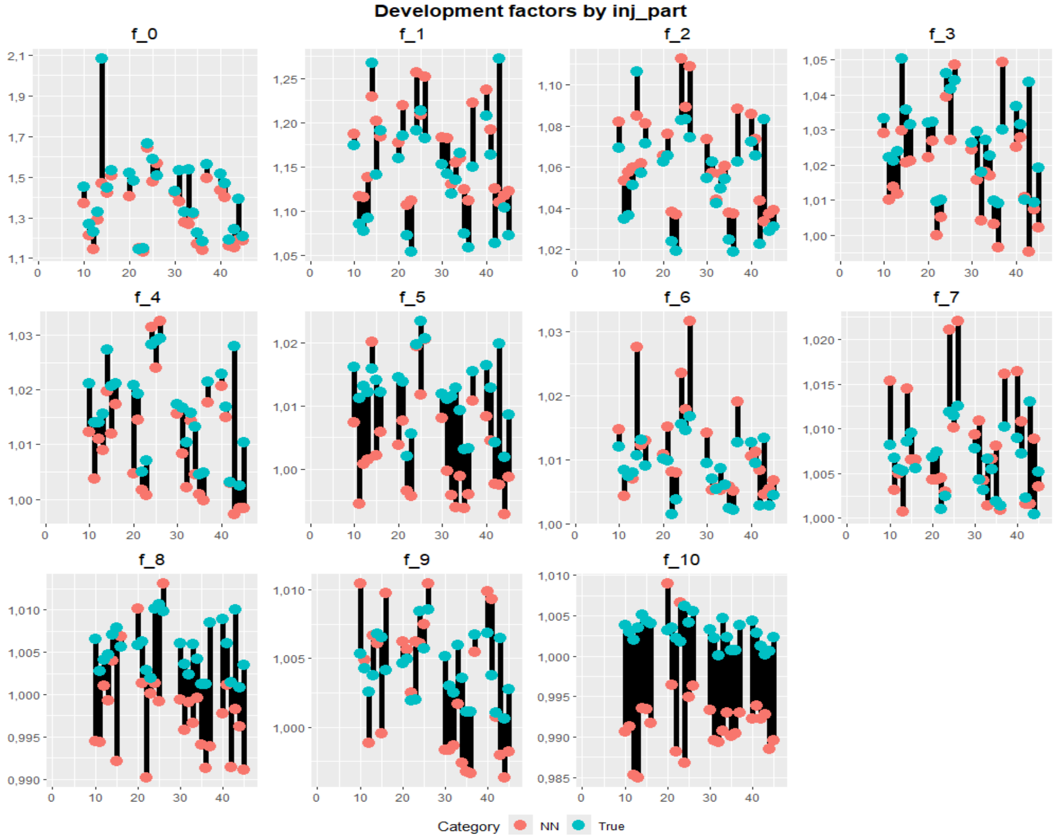
Dropout



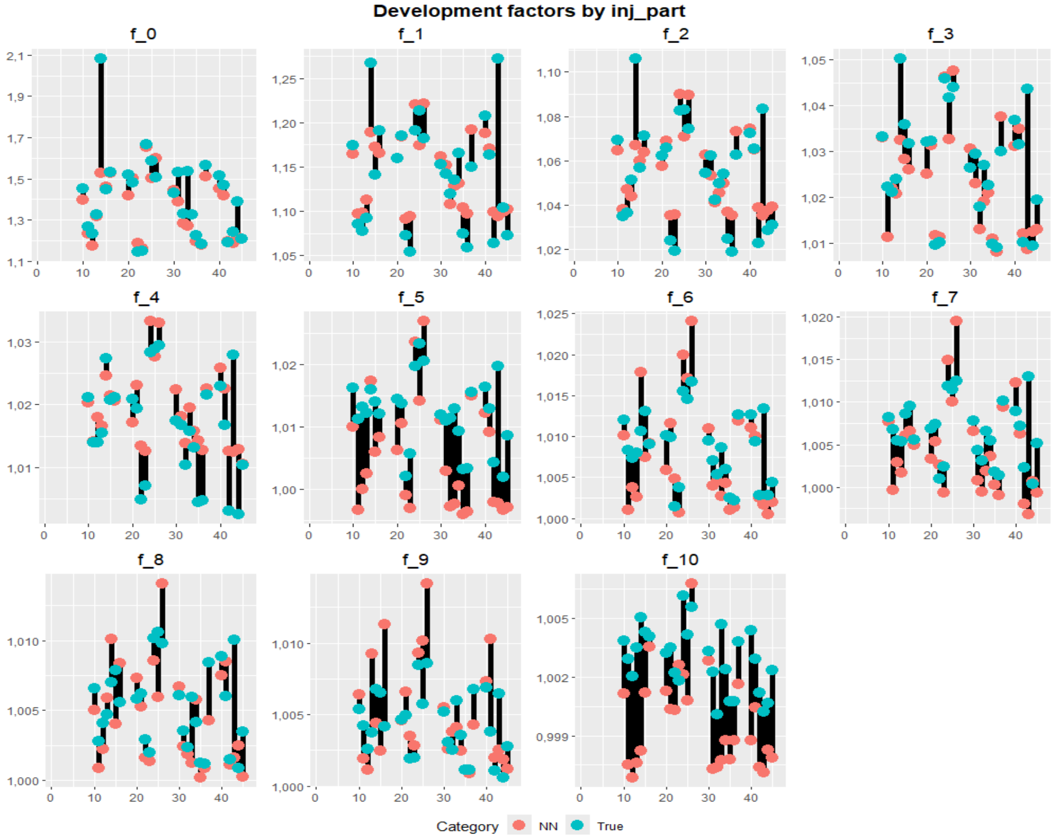
Comparison of development factors (inj_part)

- Some improvements thanks to the dropout layer can be observed

No dropout



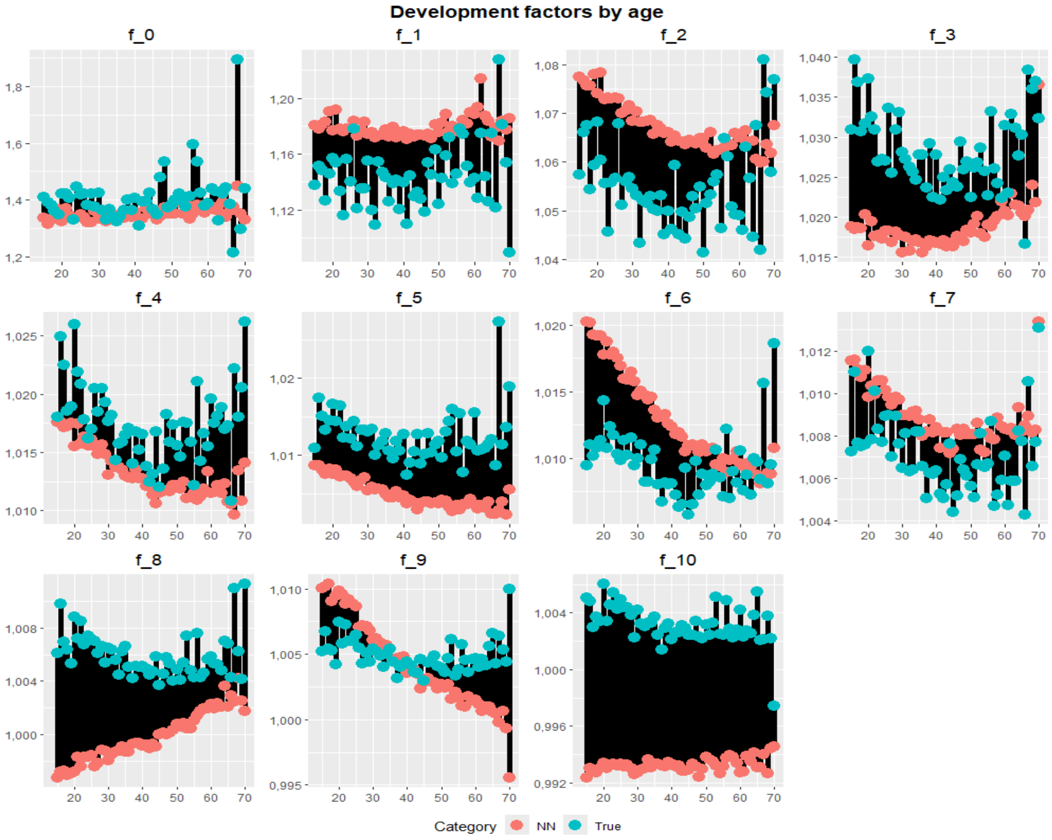
Dropout



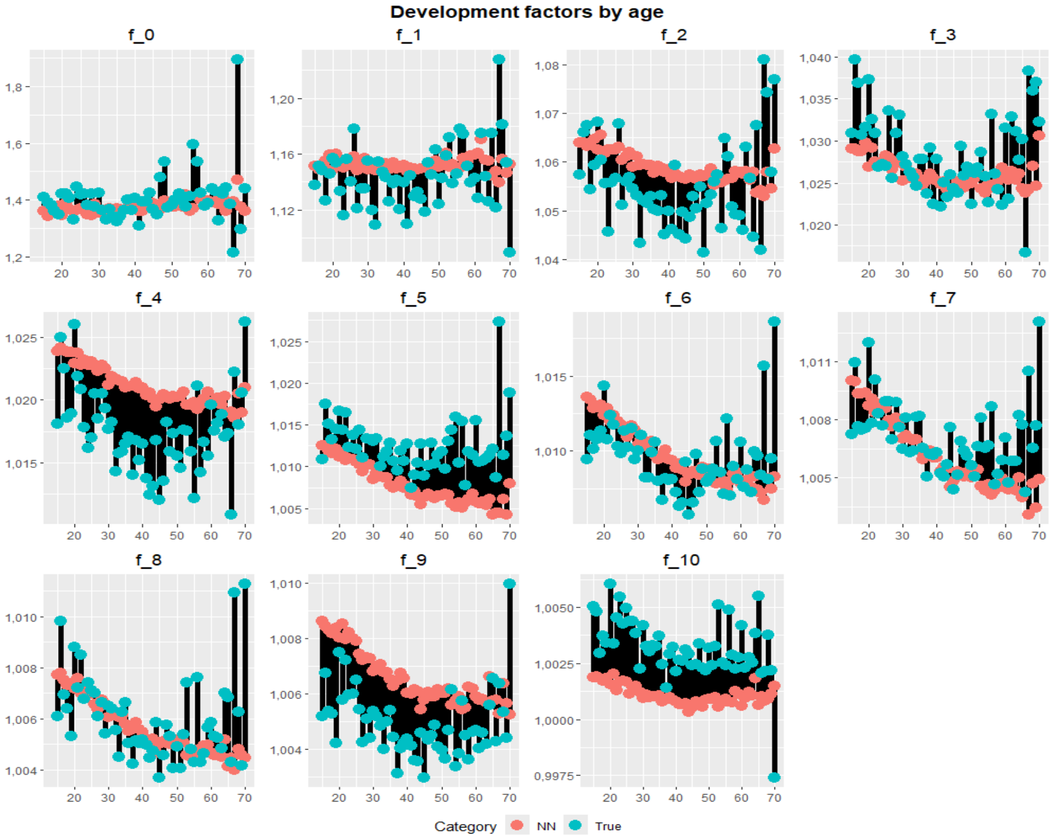
Comparison of development factors (age)

- Some improvements thanks to the dropout layer can be observed

No dropout



Dropout



8. Pros and cons of the model

Pros and cons of the applied model

Pros

- Multiple LoBs estimated together in a joint model
- The model considers a different development of various subgroups
- Part of the estimated reserve is available in a large granularity
- After the model is implemented, users just click on a button and get results, which need an interpretation

Cons

- A significant sensitivity on the random seed and some other hyperparameters
- Each development period has its own (independent) neural network model
- Lot of computation time is needed, an optimization would be nice
- No adjustments can be done by users, high data quality required

9. Overview of other explorations in neural networks models in non-life reserving

Claims Reserving and Neural Networks (2020)

- PhD. thesis by Andrea Gabrielli
- The thesis (for more details see [4]) covers (in cooperation with M. Wüthrich and R. Richman):
  - An individual claims history simulation machine (2018),
  - Back-testing the chain-ladder method (2019),
  - Neural network embedding of the over-dispersed Poisson reserving model (2020),
  - A neural network boosted double over-dispersed Poisson claims reserving model (2020),
  - An individual claims reserving model for reported claims (2021).
- The neural network model related to ODP means an integration of a classical ODP to neural networks and then it is refined through a training to reduce prediction errors

An individual claims reserving model for reported claims [4]

- The paper presents a sophisticated model, which also models zero IBNeR claims
- The neural network consists of $J + 1$ subnets modelling probability functions p_j (probabilities of a payment) and regression functions μ_j as a parameter of a log-normal distribution, i.e., the model works with more natural quantities instead of estimating "development factors"
- Probability functions and regression functions depend on features $\boldsymbol{x}$, while volatility parameters σ_j^2 do not
- The model utilizes dropouts, early-stopping and embedding weights
- initial weights are chosen before training (e.g., empirical probabilities and means of log-payments)
- Some inputs specifics:
  - the model utilizes as inputs also AY and reporting delay,
  - age is summarized into age buckets of 5 years
  - payments are categorized into 7 categories depending on their size

An individual claims reserving model for reported claims [4]

- Expected value of a payment in j-th development year is then

$$p_j(\boldsymbol{x}) \cdot \exp\{\mu_j(\boldsymbol{x}) + \sigma_j^2/2\} + p_- \cdot \mu_-$$

where constants p_- and μ_- are probability of recoveries and their expected value

- Embedding helps to find a low-dimensional representation of categorical variables, e.g., two-dimensional representation of codes cc
- Training of the model is done in two steps, first embedding weights are obtained and then the neural network is trained, having embedding weights fixed

An individual claims reserving model for reported claims [4]

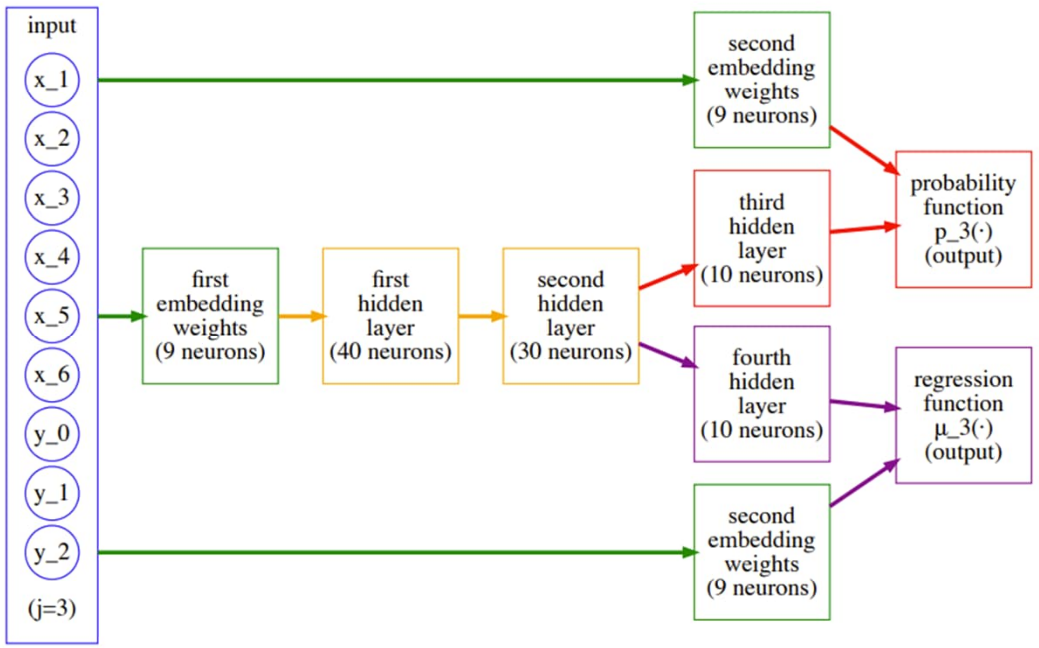


Figure 3: Model architecture of subnet $j = 3$. The five colors (blue, green, orange, red and magenta) reflect the different parts of the model.

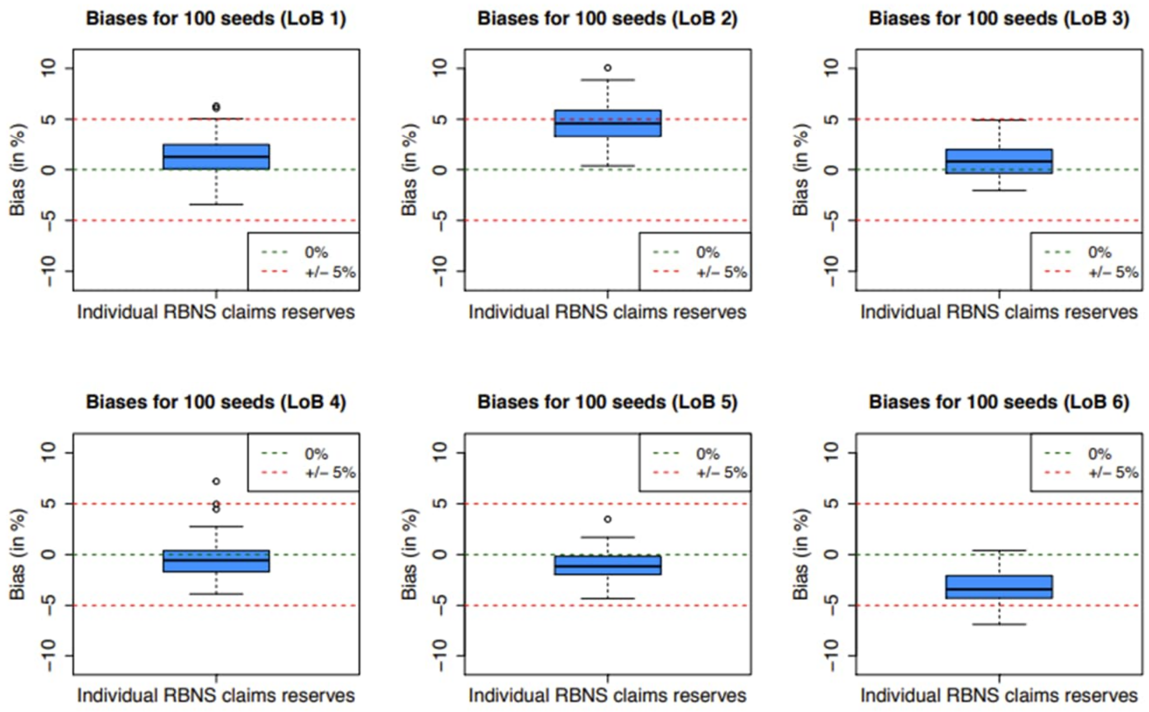


Figure 7: Boxplots of the relative biases of the individual RBNS claims reserves for 100 seeds in the second training step, for all six LoBs. The dashed green lines indicate zero bias, the dashed red lines a bias of $\pm 5\%$.

DeepTriangle: A Deep Learning Approach to Loss Reserving (2019)

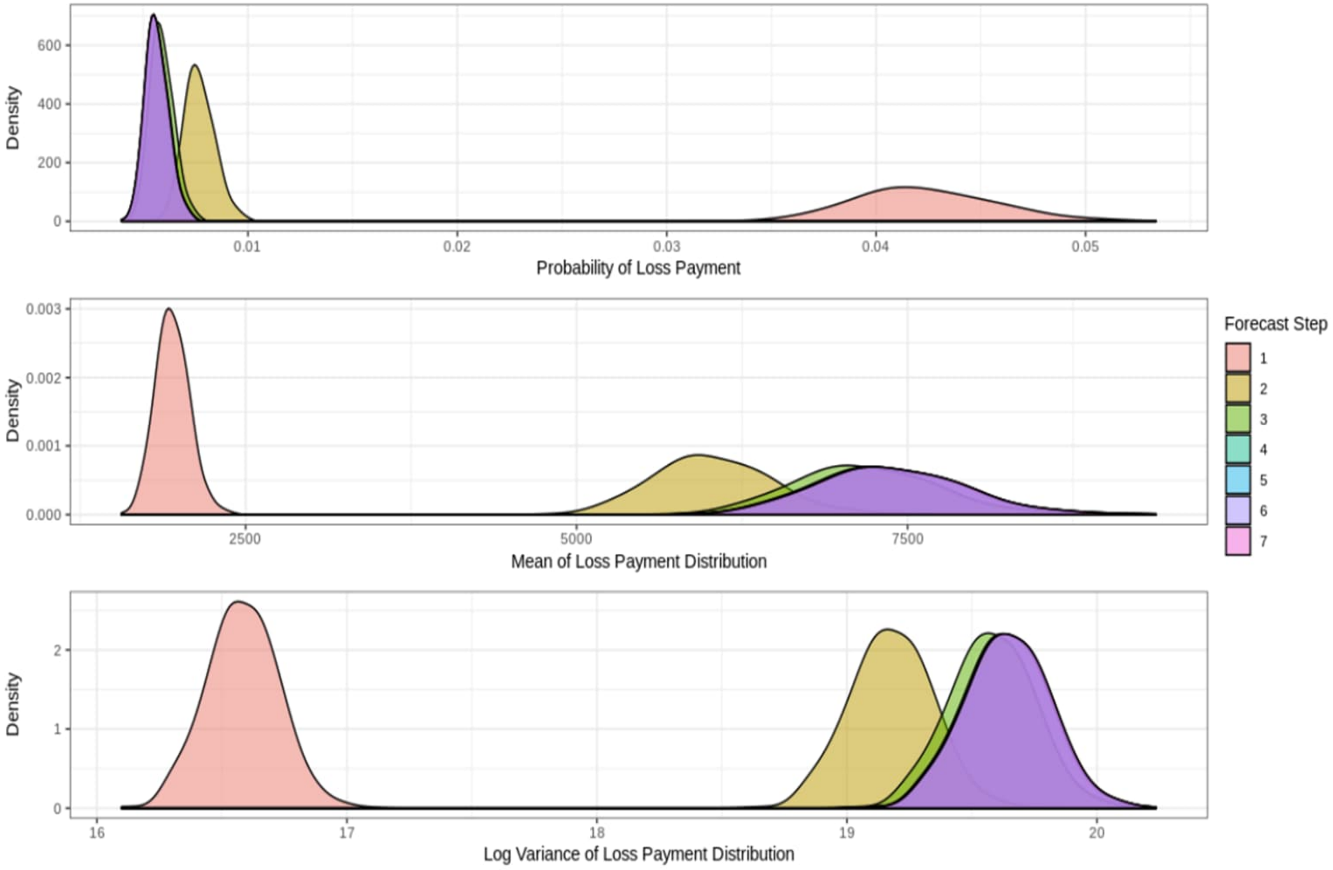
- For more details see [5]
- Author: K. Kuo
- Quite a specific case: Available paid claims, RBNS reserve and net earned premium from multiple companies in aggregated form
- Claims are normalized by earned premium and then future sequences are predicted via a neural network
- Result: Many triangles from different companies are jointly estimated with a split to single companies available (paid and outstanding claims estimated jointly, but available individually)
- Comparable performance to other methods

Collective reserving using individual claims data (2020)

- For more details see [6]
- Authors: L. Delong, M. Lindholm, M. Wüthrich
- The paper describes 6 neural networks:
    1. a network for counts of IBNR claims,
    2. a network for payments indicator and claim status of RBNS claims,
    3. a network for recoveries indicator of RBNS claims,
    4. a network for expected claims and recoveries of RBNS claims,
    5. a network for IBNR claims without any payments (zero claims)
    6. a networks for IBNR claims (non-zero claims)
- Comparable results to chain ladder, however, robustness of the model is not clear

Individual claims forecasting with Bayesian mixture density networks (2020)

- For more details see [7]
- Author: K. Kuo
- The paper introduces a framework for individual claims forecasting (only RBNS)
- The model incorporates an encoder LSTM (long short-term memory) for past payments and a decoder LSTM for generating paid loss distribution, augmented by a Bayesian neural network for an uncertainty quantification
- It does not outperform chain ladder, however, it provides a unique insight into individual reported claims via estimated distributions
- The figure on the right shows an example for one claim



Micro-level reserving for general insurance claims using a long short-term memory network (2023)

- For more details see [8]
- Authors: I. Chaoubi, C. Besse, H. Cossette, M. Côté
- Joint modelling of incremental payments and probability of their occurrence via an LSTM network
- The model considers a split to attritional and large claims to reduce variance (these parts are modelled separately)
- An application on simulated and real data
- Results suggest that the model maybe outperformed chain ladder, however, stability of the model is not clear (for example sensitivity to random seed)

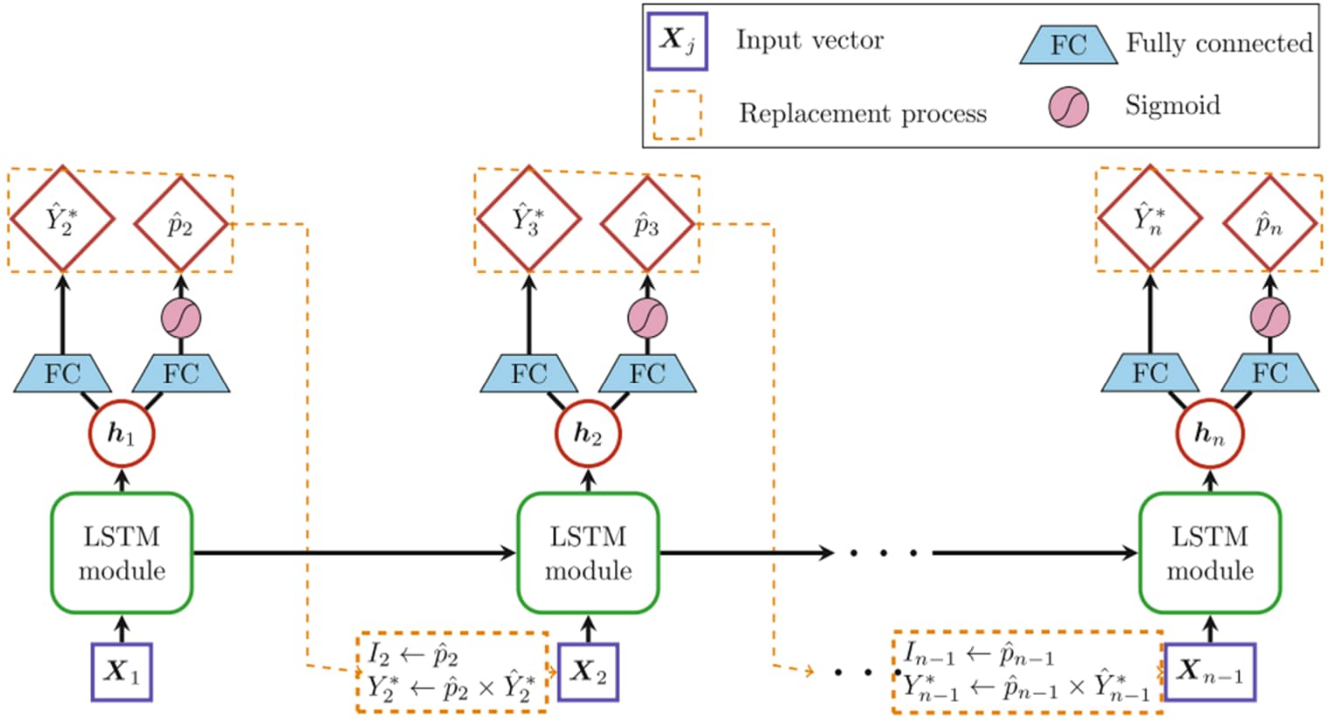


FIGURE 4 Architecture of the LSTM individual loss reserving network

Advancing loss reserving: A hybrid neural network approach for individual claim development prediction (2024)

- For more details see [9]
- Authors: J. Schneider, B. Schwab
- Data from a large industrial insurance company
- Architecture inspired by the model described in [8] with some significant changes, e.g., a decision rule is used for updating the predicted claim amounts (probability of a change in the cumulative incurred claim must exceed a predefined threshold)
- Large claims excluded, but not modelled
- Embedding of categorical variables and dropout applied
- A factor adapting the model to varying economic conditions included (GDP and inflation)
- Hyperparameters tuned by Random Search and Bayesian Search (two stages)
- Stability of the model tested by bootstrapping data, but sensitivity to random seed is not clear
- Results of the model seem promising

Advancing loss reserving: A hybrid neural network approach for individual claim development prediction (2024) – network architecture

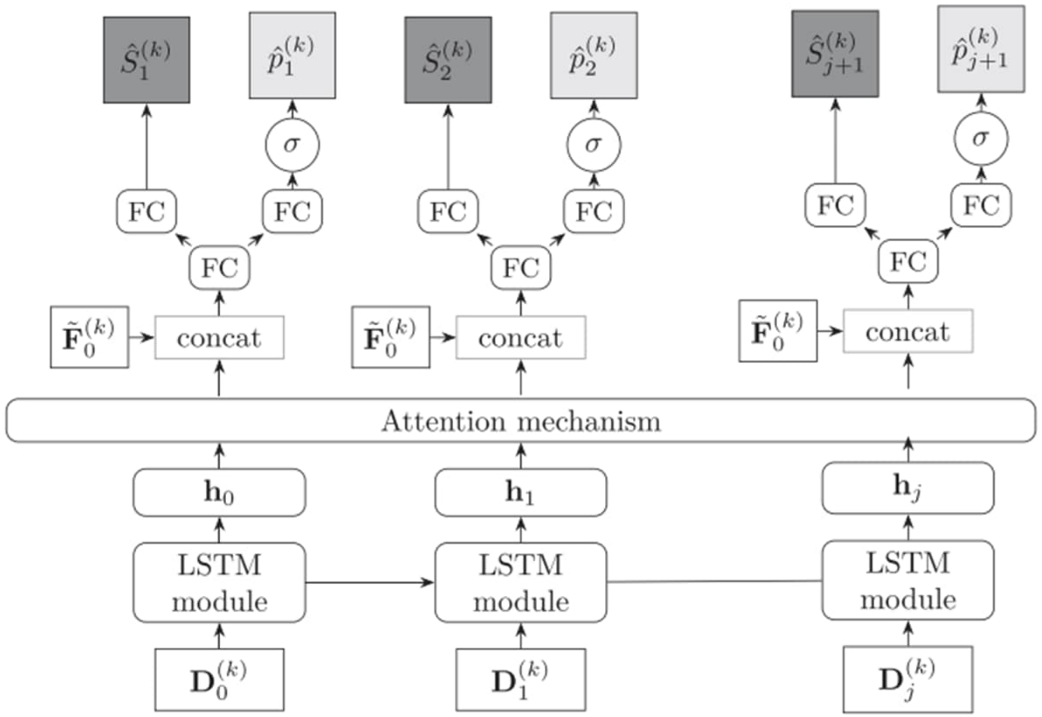


FIGURE 4 Network architecture.

10. Key ideas applied in neural network models in non-life reserving

Ideas that impressed me

- Dropout layers as a regularization technique
- Setting initial weights in a meaningful manner
- Embedding weights to have a better representation of categorical variables
- A single neural network instead of for example 11 independent neural networks
- LSTM network seems quite useful in terms of claim triangles or time series
- Possibility of algorithmic tuning of hyperparameters

11. Conclusion

Conclusion

- The applied model suffers from a significant sensitivity to random seed (and some other parameters, which are subject to a right calibration)
- Because of the volatility, the model does not outperform chain ladder
- Nevertheless, the model has served as an inspiration for more advanced models, and it quite nicely shows, what to be aware of when applying neural networks in non-life reserving
- Models in [4] and [9] seem quite robust
- Neural networks have a large potential in non-life reserving

Literature

Literature

- [1]: Gabrielli, A., & Wüthrich, Mario V. (2018). An Individual Claims History Simulation Machine. *Risks*, *6*(2), 29. Available at https://doi.org/10.3390/risks6020029
- [2]: Wüthrich, Mario V., & Merz, Michael (2008). Stochastic Claims Reserving Methods in Insurance, ISBN 978-0-470-72346-3
- [3]: Wüthrich, Mario V. (2018). Neural Networks Applied to Chain-Ladder Reserving. Available at SSRN: https://ssrn.com/abstract=2966126 or http://dx.doi.org/10.2139/ssrn.2966126
- [4]: Gabrielli, A. (2020). Claims Reserving and Neural Networks (PhD Thesis). Available at https://doi.org/10.3929/ethz-b-000445511
- [5]: Kuo, K. (2019). DeepTriangle: A Deep Learning Approach to Loss Reserving. Available at https://www.mdpi.com/2227-9091/7/3/97
- [6]: Delong, L., & Lindholm, M., & Wüthrich, Mario V. (2020). Collective Reserving using Individual Claims Data. Available at https://www.tandfonline.com/doi/full/10.1080/03461238.2021.1921836?scroll=top&needAccess=true
- [7]: Kuo, K. (2020). Individual Claims Forecasting with Bayesian Mixture Density Networks. Available at https://arxiv.org/abs/2003.02453
- [8]: Chaoubi, I., & Besse, C., & Cossette, H., & Côté, M. (2023). Micro-level reserving for general insurance claims using a long short-term memory network. Available at https://onlinelibrary.wiley.com/doi/full/10.1002/asmb.2750
- [9]: Schneider, Judith C., & Schwab, B. (2024). Advancing loss reserving: A hybrid neural network approach for individual claim development prediction. Available at https://onlinelibrary.wiley.com/doi/10.1111/jori.12501
- [10]: My code is available at: Neural-networks-NL-reserving-/NN code at main · HaskeerCZ/Neural-networks-NL-reserving- · GitHub